오픈 API를 이용한 웹 페이지 설계

2021년 12월 14일

2021년 2학기 소프트웨어시스템실습

담당교수: 최성종

이름 이승원

Kkod3327@office.uos.ac.kr

서울시립대학교 전자전기컴퓨터공학부

**요약**

이 보고서에서는 2021년 2학기 소프트웨어 시스템 실습 과목을 수강해 배운 것들을 기반으로 재난 상황에 도움이 되는 웹 페이지를 만드는 과정을 다루고 있습니다.

Ajax, JSON등의 개념을 간략히 설명하고 이를 이해할 수 있는 예제 코드를 분석해 본 후, 자신이 재난 관련 오픈 API를 이용해 재난 상황에서 대처하는 것에 도움을 줄 수 있는 정보를 네이버 지도로 표시해 주는 웹 페이지를 만들었습니다.

이 웹 페이지를 제작하는 데 고려한 요구 사항과 이것들을 실현할 방법, 이로써 만들어진 페이지 등을 확인할 수 있으며 Html 포맷으로 만들어진 웹 페이지는 github를 통해 업로드 되어 접근 권한이 있는 누구나 이용할 수 있습니다.

차 례

[1. 주요 개념 1](#_Toc90377032)

[1.1. JQuery – AJAX 1](#_Toc90377033)

[1.2. JSON 1](#_Toc90377034)

[1.3. GeoJSON 1](#_Toc90377035)

[2. 예제 코드 분석 1](#_Toc90377036)

[2.1. 데이터 레이어 예제 1](#_Toc90377037)

[2.2. 재난 경보 안내 페이지 실습 1](#_Toc90377038)

[3. 웹 페이지 설계 7](#_Toc90377039)

[3.1. 요구사항 7](#_Toc90377040)

[3.2. 설계 및 구현 8](#_Toc90377041)

[4. 결과 14](#_Toc90377042)

[4.1. 페이지 14](#_Toc90377043)

[4.2. 수정&보완할 사항 15](#_Toc90377044)

[4.3. 의의 16](#_Toc90377045)

[5. 결과 17](#_Toc90377046)

[5.1. 부록 17](#_Toc90377047)

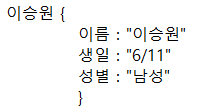
# 주요 개념

## JQuery – AJAX

JQuery는 자바 스크립트의 라이브러리이고 AJAX는 서버를 호출하고 응답을 받아 페이지를 업데이트 할 수 있는 기능이다.

## JSON

객체 혹은 구조체의 정보를 주고 받을 때, 이것을 효과적으로 전하는 방법은 여러가지가 있다. 예를 들어 <그림1>과 같은 객체가 있다고 하자.



<그림 1> 객체 예시

이 정보를 “이승원”,”6/11”,”남성” 이렇게 보낼 수도 있지만, 키워드와 값의 쌍을 통해 "이름","이승원","생일","6/11","성별","남성" 이런 식으로 보낼 수도 있을 것이다. JSON은 후자의 방법으로 보내는 정보전달 형식으로, 프로퍼티의 키워드를 검색해 필요한 값을 추려낼 수 있는 장점이 있다.

## GeoJSON

GeoJSON은 1.2의 JSON의 한 형식으로, 지리 좌표계의 점을 기반으로 다양한 지형을 표현하는 데이터의 표준이다. 위도와 경도 표기와 점, 선, 다각형 등 다양한 기하학적 정보를 전달하는 것에 특화되어 있다.

# 예제 코드 분석

## 데이터 레이어 예제

 텍스트이(가) 표시된 사진

자동 생성된 설명

<그림2> 예제 1 전문

1. <head>

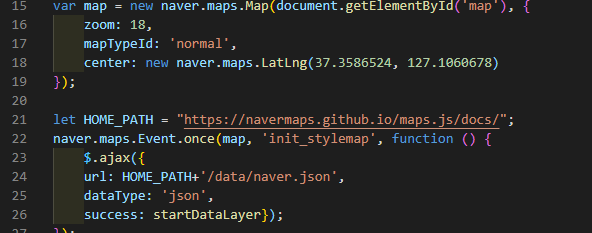
이전 실습들에서 작성했던 통상적인 html 구조들과 달리 이 코드에서 눈 여겨 볼 곳은 8번과 9번 줄이다.



<그림3> head

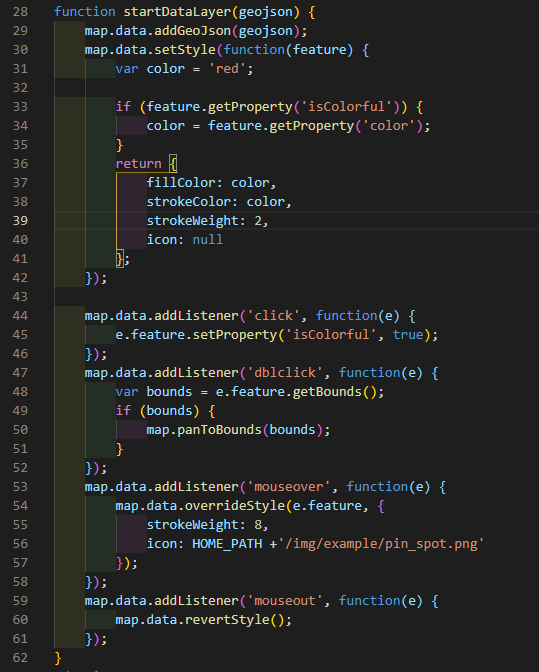
8번은 JQuery 구문을 사용하겠다는 것을 미리 알리는 것이고 9번은 Naver 사의 MAPs open API를 활용하기 위한 클라이언트 ID다. J Query로 요청했을 때 지급받은 클라이언트 ID가 없다면 거절당하게 되기 때문에, 데이터를 요청해 사용하기 위해선 사전에 사용 신청을 하고 발급 받는 절차를 거쳐야한다.

1. <body>



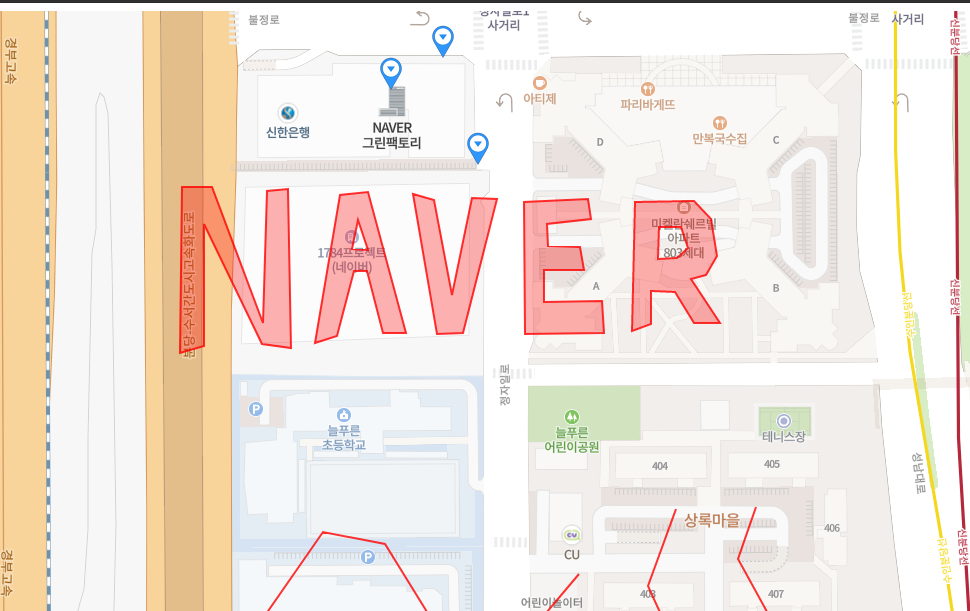
<그림4>body(1)

<그림4>의 15번 줄은 naver.maps에서 제공하는 Map 클래스의 생성자다. 18번 줄에서 지도의 중심이 될 위도와 경도를 지시하고 있다. 한편, 23번 줄에서는 1.1에서 언급한 ajax 구조를 살펴볼 수 있다. url은 네이버 맵의 github 내에서 예시 구조체를 가리키고, dataType에서 이것이 json 구조체임을 알 수 있다. 26번 줄의 success는 ajax 요청이 성공적으로 처리되었을 때 실행할 함수를 나타낸다.



<그림 5> body(2)

<그림5>에서 startDataLayer 함수가 geojson을 받아 실행되는 것을 알 수 있다. ajax의 요청 결과로, 네이버 맵 허브에서 geojson형태의 데이터를 얻고, 29번 줄에서 이를 maps.data 객체의 addGeoJson 메소드를 통해 불러온다. 30번부터 42번 줄은 데이터의 초기 상태(표시할 색)를 설정하고 44번부터 61번줄은 각각 클릭, 더블클릭, 마우스 올리기, 마우스 벗어나기에 대한 이벤트 리스너들의 생성자이다.



<그림6> 예제 코드 실행 결과

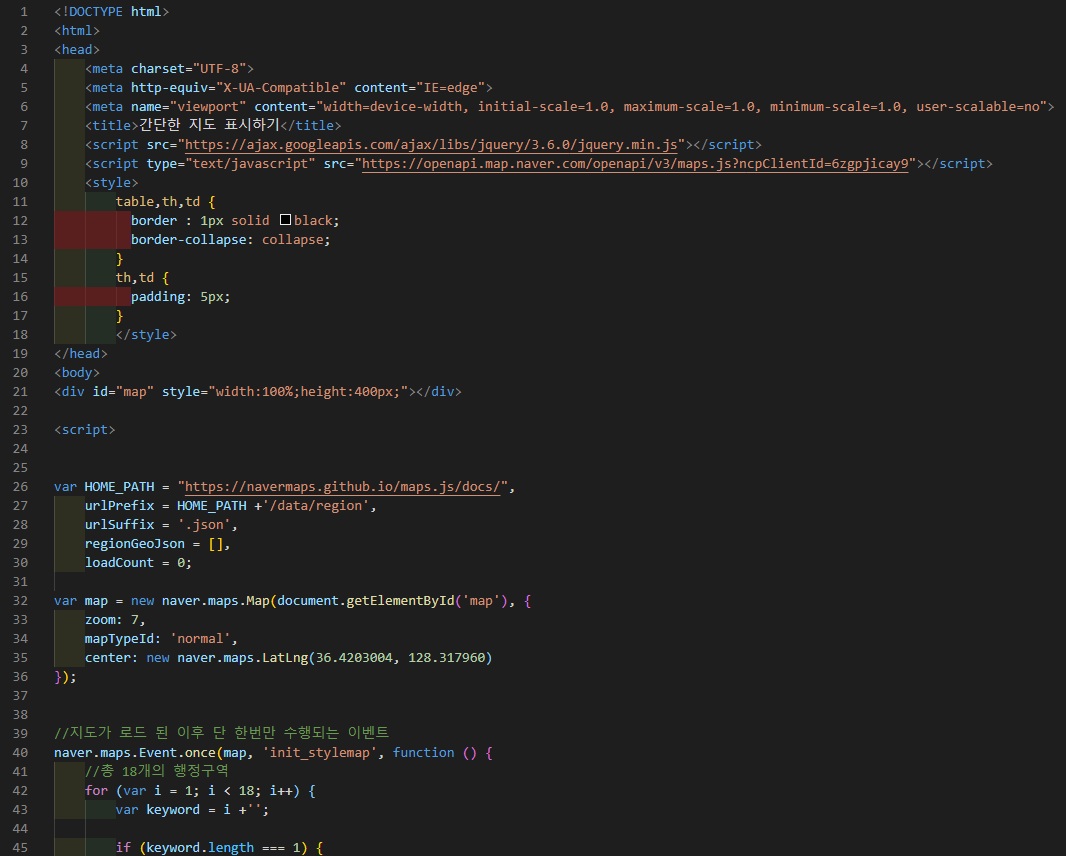
<그림6>은 지도 부분과 데이터 부분으로 분할되어 있다. 우리는 이것을 각각 Map 객체와 map.data로 제어할 수 있다. <그림6>에 나온 지도 위의 NAVER라는 글자와, 빨간 줄, 그리고 몇몇 파란 마크들이 우리가 ajax로 요청한 데이터(GeoJson 구조였던)에 포함되어 있었을 것이다. 데이터에 포함시켰던 리스너들의 콜백함수는 <표1>에서 확인할 수 있다.

|  |  |  |
| --- | --- | --- |
| 이벤트 | 표시 | 설명 |
| 마우스 올리기 (Mouse over) |  | 스타일을 override 한다. 구체적으로는 strokeweight(두께)를 두껍게 만든다. |
| 클릭 (Click) |  | 클릭한 객체의 ‘isColorful’ 프로퍼티 값을 true로 만든다. IsColorful은 객체의 색을 표시할 지를 결정하는 프로퍼티다. |
| 더블 클릭  (Double click) |  | 더블 클릭한 지점을 기반으로 확대한다. 다른 리스너들과는 달리, 이 콜백 함수는 map.data가 아닌 map의 메소드이다. |
| 마우스 내리기  (Mouse out) |  | 마우스를 올렸을 때 객체의 테두리 두깨를 override 했던 것을, 마우스를 내렸을 때 원상복귀 시키도록 한다. |

<표1> 리스너 & 콜백 함수 1

## 재난 경보 안내 페이지 실습

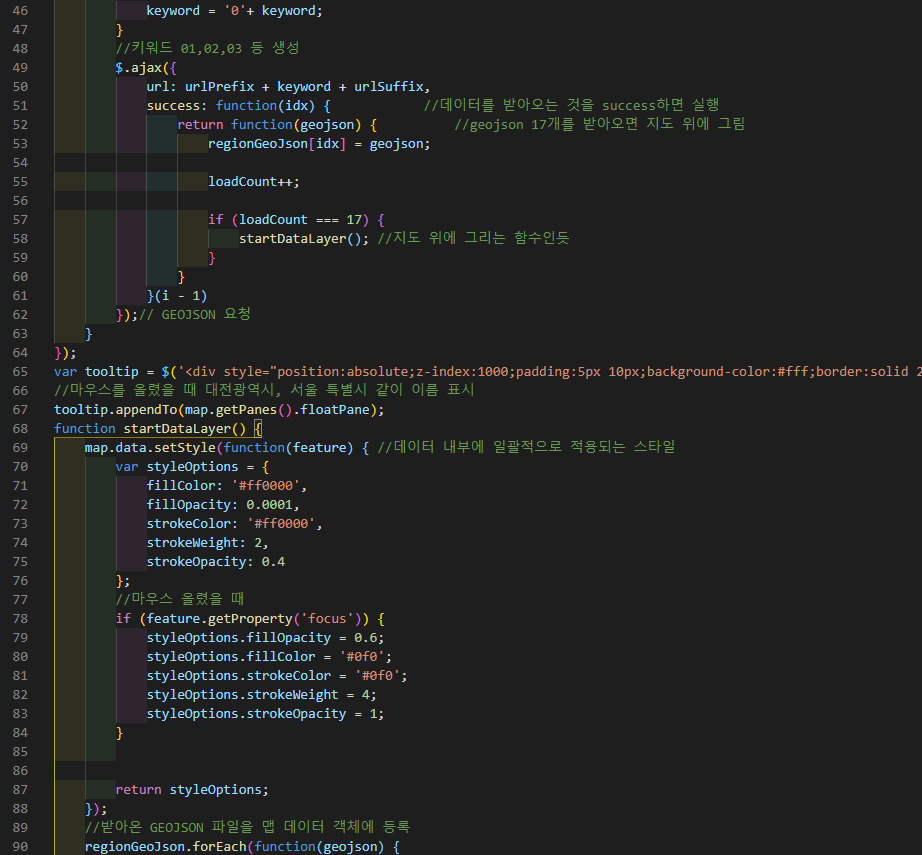
1. 전반부(ID, URL)



<그림 7>

2.1에서 언급했듯이 8번 줄은 JQuery, 9번 줄은 클라이언트 ID다. 10~18번 줄은 table 객체에 대한 스타일을 미리 지정하고 있다. 26번부터 30번 줄은 가져와야 할 데이터가 HOME\_PATH + ‘/data/region’+ keyword + ‘.json 형태의 url 주소를 가져서 반복문을 통해 여러 데이터를 가져오기 위해 작성됐다. 32~36번 줄은 2.1에서 다룬 Map 개체 생성자임을 확인할 수 있고 40번 줄부터 로드 된 후 한번만 실행되는 once 메소드가 실행할 함수가 작성되어 있다.

1. 중반부(ajax, tooltip, Data Layer)



<그림 8>

42번 줄부터 반복문이 작성되어 있다. Ajax를 통해 데이터를 받아오는데, 그 주소는 26~30번 줄에서 작성된 형식을 이용한다. 총 18번의 반복을 통해 18개 행정 구역의 geojson 데이터를 받아 온 후 regionGeoJson 배열에 추가한다.(~53번 줄) 이후 모든 데이터를 받게되면 startDataLayer 함수를 실행한다. 67번은 툴팁 부분으로, appendTo는 부모 노드에 HTML을 추가하는 메소드고, getpanes는 map 객체가 오버레이 할 수 있는 창 요소 집합을 반환한다. 이때, floatpane은 모든 창 위에 표시될 창을 뜻한다. 즉 툴팁이 지도 혹은 데이터 영역보다 위에 표시되게 되는 것이다. 68번 줄부터 작성된 starDataLayer 함수는 말 그대로 데이터 영역을 생성하는 함수다. 69번 줄의 setStyle 메소드를 통해 GeoJson 객체 테두리 선의 굵기, 색상 등이 제시된다. 여기까지의 진행으로 지도 및 geojson 데이터 레이어가 추가된 <그림9>를 얻을 수 있다. <

지도이(가) 표시된 사진

자동 생성된 설명

<그림9> 표시된 지도

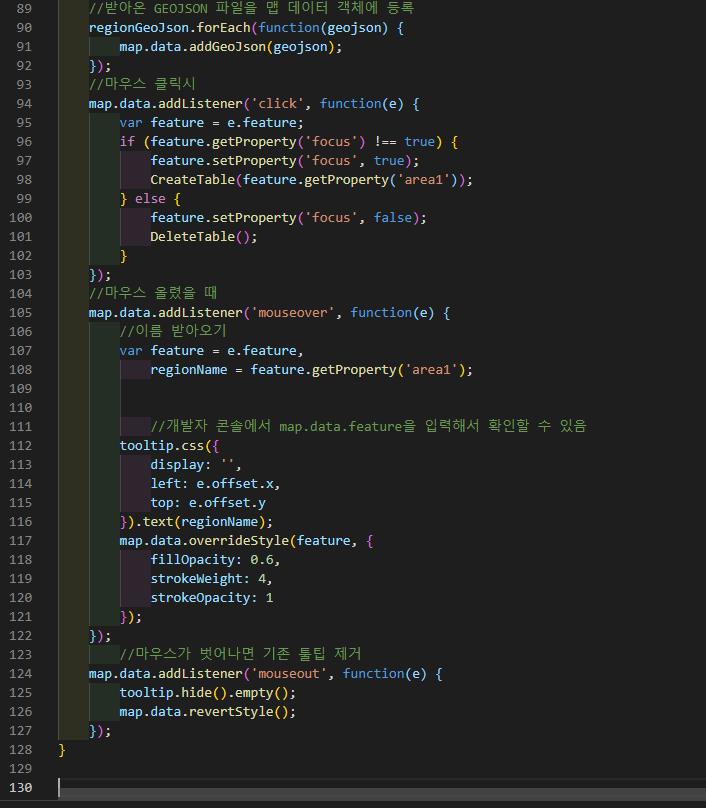
한편, 78번부터 85번까지는 객체의 프로퍼티 값 중 ‘focus’가 true 일 때 실행되는 것으로, focus는 후에 96번 줄에서 마우스 올리기 리스너에 의해 상태가 변화한다. 즉 이 부분은 마우스를 올렸을 때 객체의 스타일을 변화시키는 부분이다.



<그림10> 마우스 올림

<그림10>에서 마우스를 올렸을 때 생긴 변화를 확인할 수 있다. GeoJson의 속성이 변하고, 툴팁이 가장 위에 표시된 것이 보인다.

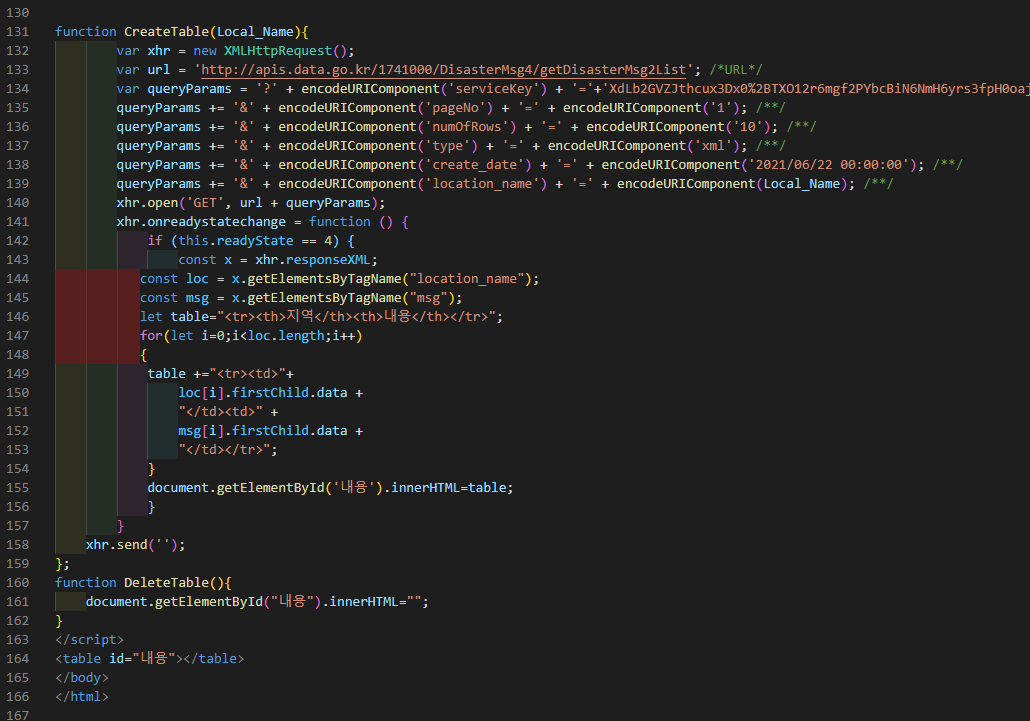
1. 후반부(리스너)



<그림11>

94번 줄부터 127번까지는 리스너에 대한 부분이다. 마우스 클릭시 객체의 focus 프로퍼티가false라면 true로 만든다. 이 때 focus가 true가 되면 78~85번 줄에 의해 객체에 표시되는 스타일이 바뀐다(녹색으로 칠해진다). 그리고 CreateTable 함수를 실행하는데 이것은 잠시 후에 살펴보겠다. 만약 이미 true라면, focus를 false로 되돌리고 DeleteTable 함수를 실행한다. 객체에 마우스를 올렸을 때는 108번 줄에서 객체의 area1(시/도 이름)를 가져오고 이를 116번 줄에서 툴팁으로 표시한다. 또한 마우스를 올리고 내릴 때도 스타일을 변화시킨다(붉은 색으로 칠해진다). 위의 리스너들과 콜백은 <표2>에 정리되어 있다.

1. 후반부 (표 작성 & 제거)



<그림 12>

131번부터 159번 줄은 CreateTable 함수를, 160번과 161번 줄은 DeleteTable 함수를 이루고 있다. CreateTable 함수는 132번 줄의 url로, 133번의 공공데이터 포털에서 제공하는 재난 문자 발송 현황 API에 대해 미리 발급받은 클라이언트 ID를 이용해 요청을 보낸다. 134줄부터 139번은 필요한 요소들에 대한 키워드와 요청하는 정보의 구체적인 내용을 담고 있다. 예를 들어, 135번 줄은 API가 제공하는 재난 문자 리스트에서 1페이지의 정보들을 요구하고 136번 줄은 10개의 정보를 요구하고 있다. 이렇게 요청한 후에 140번 줄에서 XMLHttpRequest 클래스의 .onreadystatechange 메소드에 의해 요청이 완료된 후 다음 명령을 이행한다. 142번부터 155번은 내용을 토대로 테이블을 작성하는 부분이다. 재난 문자가 발송된 지역명과 내용을 표로 나타낸다. 이 표는 164번 줄에 있는 id가 “내용” 인 테이블 객체에 작성된다. 마지막으로 DeleteTable 함수는 이 테이블 객체의 내용을 “ ”로 변경함으로써, 이미 작성된 테이블을 지우는 효과를 얻는다.

|  |  |  |
| --- | --- | --- |
| 이벤트 | 표시 | 설명 |
| 마우스 올리기 (Mouse over) | 지도이(가) 표시된 사진  자동 생성된 설명 | 스타일을 override 한다. 색을 채우고, 툴팁을 표시한다. |
| 클릭 (Click) |  | 클릭한 객체의 ‘focus’ 프로퍼티 값을 true로 만든다. 또한 재난 문자에 대한 표를 작성한다. |
| 다시 클릭  (Click again) |  | 클릭하기 전의 상태로 돌아가고, 작성됐던 표를 제거한다. |
| 마우스 내리기  (Mouse out) |  | 마우스를 올렸을 때 객체의 테두리 두깨를 override 했던 것을, 마우스를 내렸을 때 원상복귀 시키도록 한다. |

<표2> 리스너 & 콜백 함수2

# 웹 페이지 설계

## 요구사항

1. 문제 제시

공공 데이터 포털의 재난 관련 정보를 활용하여, 재난 상황에 대한 대처에 도움이 되는 웹 페이지를 만들어야 한다. 예제에서는 공공 데이터 포털에서 제공하는 API를 통해 지역별 재난 문자(주로 코로나) 발송 현황을 알아봤다. 그러나 재난은 질병만이 있는 것이 아니며, 언제 어디서나 갑작스럽게 겪을 수 있는 것이다.

1. 사용 시나리오

2-a) 거주 지역의 코로나 현황에 대한 정보를 얻기 위해 페이지를 이용한다.

2-b) 지진을 느끼고 빠르게 대피할 장소를 찾는다.

2-c) 폭우가 오래 지속되어 홍수가 발생해 대피소로 사용할 건물을 조회하고, 침수된 집에서 안전한 곳으로 이동한다.

1. 기능적 요구사항

3-a) 공공데이터 포털에서 제공하는 API를 이용해, 이용자가 원하는 지역의 데이터를 수신한 후 발생 시간 순서로 표로 정리해 보여준다.

3-b) 행정 안전부에서 제공하는 지진 발생시 옥외대피장소 안내 서비스를 이용해, 네이버 지도에 표시해준다.

3-c) 한국국토정보공사에서 제공하는 API를 활용해 침수 발생 시 이용가능한 대피소를 사용자의 현재 위치에서 가장 가까운 곳으로 안내해준다.

1. 비 기능적 요구사항

이 웹 페이지는 재난 상황에서의 활용을 목적으로 하므로, 응답이 신속하게 이루어져야 한다. 또한 크롬 등의 대부분의 인터넷 사용 환경에서 정상적으로 동작해야 한다.

## 설계 및 구현

1. 코드의 주요 구조

1-a) 탭 메뉴 구성



<그림13>

탭 메뉴를 통해 재난을 선택하면 그에 맞는 정보가 안내된다. 각각의 탭에는 네이버 지도 객체가 배정되어 있으며 이 지도는 클라이언트가 로드 했을 때 공공데이터 포털의 API에 접속해서 필요한 데이터를 얻어 네이버 지도에 표시해준다. 또한 id명이 “내용”으로 설정된 div 블록에서는 데이터 로드가 완료되면 ‘로딩 중’이라는 innerHTML이 바뀌면서, 데이터의 로딩이 잘 처리되었는지 피드백 받을 수 있다.

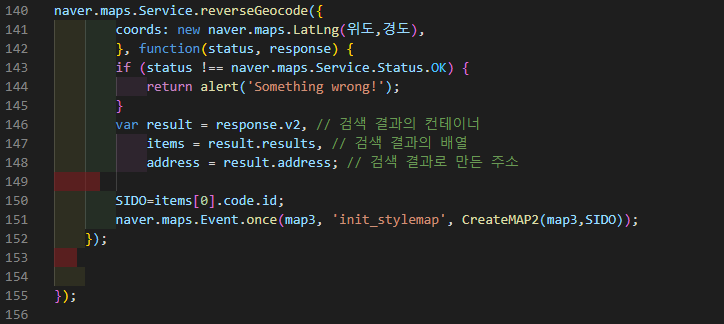
1-b) 현재 위치 파악 및 지도 생성

텍스트이(가) 표시된 사진

자동 생성된 설명<그림14>

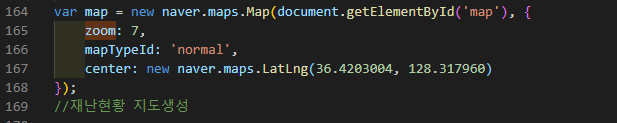
112번 줄에서 getCurrentPosition 메소드를 통해, 접속과 동시에 유저의 현재 위치 정보를 PC라면 IP주소를 이용해, 모바일이라면 GPS를 이용해 받는다. 이 정보는 전역 변수 ‘위도’ 및 ‘경도’에 저장되며, 이 현재 위치를 이용해 Map 객체를 현재 위치 기준으로 생성하고 마커를 생성한다.

1-c) 도로명 주소, 행정 코드 획득



<그림15>

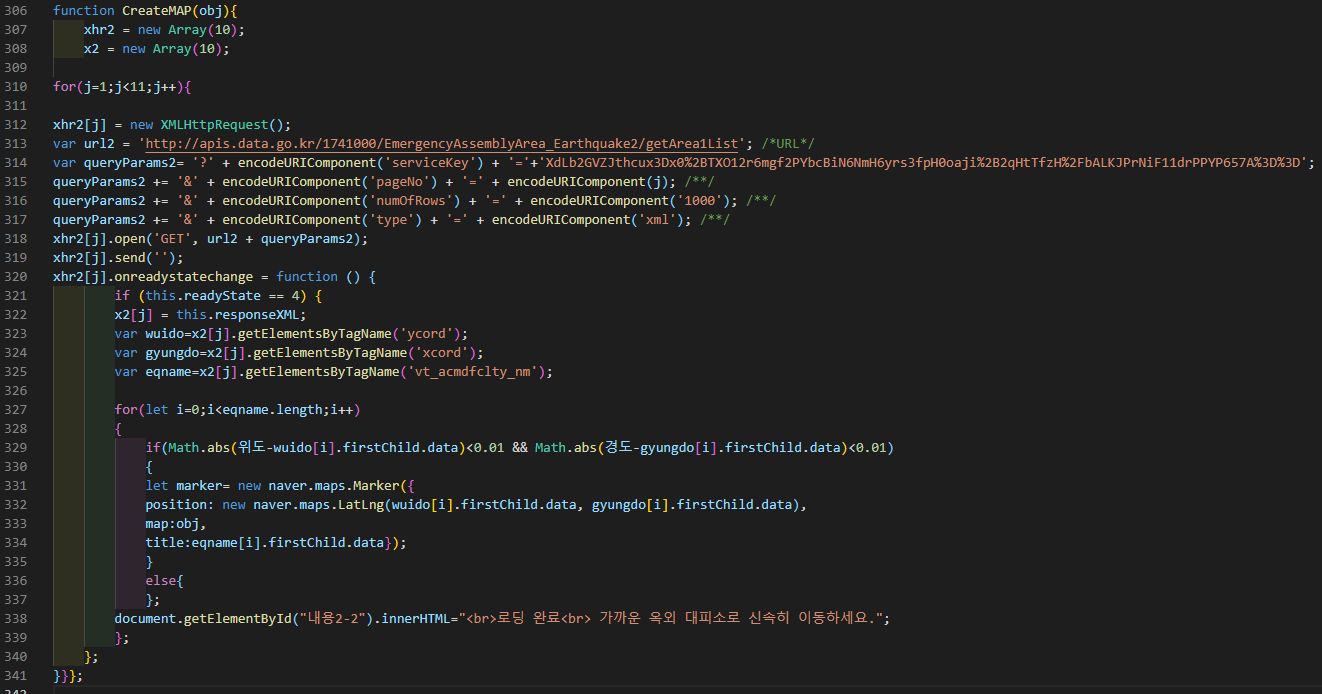
현재 위도와 경도를 기반으로 주소, 법정동코드(행정표준코드[4]) 등을 획득한다. 151번 줄에서는 3번 지도의 로딩이 끝나면CreateMAP2라는 함수를 실행하는데, 이 때 CreateMAP2는 요청할 API의 데이터를 SIDO라는 변수에 저장된 행정표준코드를 이용해 얻는다. map3은 이때 얻은 데이터로 마커를 3번 지도 객체에 생성하기 위한 참조변수다.



<그림16>

<그림>에서는 1번 지도의 경우, 우리나라 전도를 기준으로 동작하기 때문에 현재 위치가 필요 없는 것을 확인할 수 있다.

1-d) 탭 2번 메뉴. 옥외 대피소 마커 생성

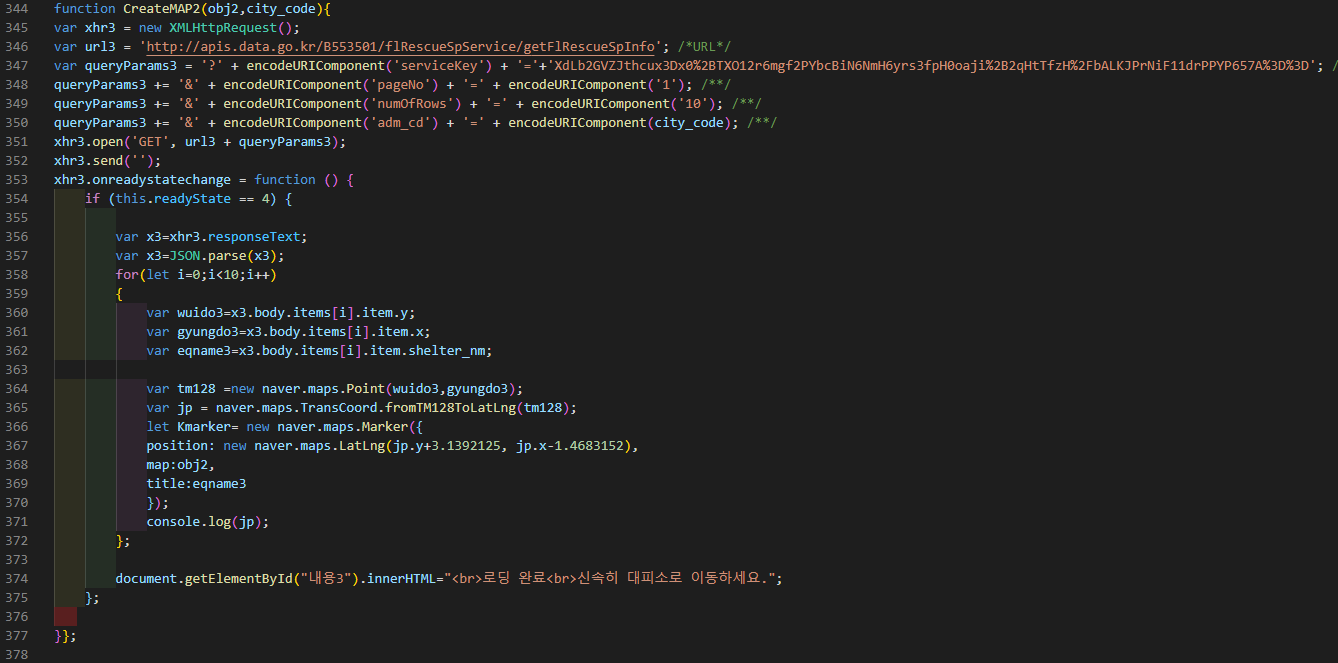


<그림17>

xhr2와 x2는 각각 배열로 선언된다. 이 10크기의 배열 둘은 반복문에 의해서 각각 XMLHttpRequest 객체와Xhr2[j].responseXML 문서로 채워진다. 1번의 시행에 1000개의 데이터 요소를 수집한다. 이를 모두 지도에 표시하는 것이 아니라, 생성된 XML 데이터에서 wuido는 위도를, gyungdo는 경도를, eqname은 시설의 이름을 각각 저장하고 ‘위도’ 변수와 wuido의 차이가 0.01이내이면서 동시에 ‘경도’ 변수와 gyungdo의 차이가 0.01 일 때 지도에 추가한다.

‘위도’와 ‘경도’ 변수가 1-b)에서 얻은 현재 위도와 경도이므로, 현재 위치를 기준으로 위도와 경도 0.01 이내인 곳에 해당하는 옥외 대피소에 대해서만 마커를 생성하게 된다. Eqname은 이 마커의 title 속성이 되어 마우스를 올릴 경우 정보를 확인할 수 있다.

1-e) 탭3번 메뉴, 침수 대피소 마커 생성



<그림18>

CreateMAP2 함수는 map3와 SIDO ( 현재 위치한 지역의 행정표준코드 )를 받아 대피소에 대한 데이터를 받아온다. 그런데 이 API는 데이터를 Text형식으로 제공하므로 356번 줄에서 responseText메소드를 이용해 Text로 표시한 후, JSON.parse 메소드를 이용해 다시 JSON 구조로 만들었다. 이렇게 얻은 데이터의 x, y좌표는 위도&경도를 나타내는 것이 아니라 tm128이라는 좌표계로 이루어져 있다. 그래서 이것을 naver.maps.Point 메소드로 포인트 객체로 만든 뒤 네이버 서브 모듈인 geocorder에서 제공하는 fromTM128TOLatLng 메소드[5]를 이용해 위도와 경도로 바꾼 후 마커를 생성해, map3에 추가하게 된다.

1. 텍스트, 모니터, 화면, 스크린샷이(가) 표시된 사진

   자동 생성된 설명필수 라이브러리

<그림19>

이전의 다른 부분들은 2.2예제에서 설명했으므로 생략하겠다. 차이점은 10번 줄이다. Geocoder는 위도&경도, UTM-K, TM128 등 다양한 표현 방식의 좌표들을 다루는 네이버 지도의 서브 모듈이다. 이 기능은 현재 위치에서 위도와 경도를 얻고, 이것에서 네이버 지도를 이용해 시/도 등의 위치 정보를 얻고 다시 이 시/도 정보로 적절한 데이터를 API를 이용해 데이터 포털에 요구하는 식으로 활용했다.

# 결과

## 페이지

지도이(가) 표시된 사진

자동 생성된 설명

<그림>tab1 메뉴

1번 탭 메뉴에는 이전에 만들어 둔2.2예제를 넣었다. 구조와 기능은 동일하나 다른 탭 메뉴를 선택하게 되면 내용을 숨기도록 설정했다. 이용자가 수직으로 스크롤 하는 것보다 탭을 이용해 모든 내용이 한눈에 내용이 전달되어야 한다고 생각했다.

지도이(가) 표시된 사진

자동 생성된 설명

.<그림>tab2 지진 발생 시 야외 대피 장소

2번 탭 메뉴를 만드는 데 활용한 API에서는 이용자의 주소지나 행정 구역 등의 정보를 요청 변수에 할당하지 않았다. 문서의 데이터는 총 10,400여 개였는데 이것은 정렬조차 되어있지 않았다. 이를 해결하기 위해서, 한 번에 처리할 수 있는 최대치인 1000개의 노드를 가져오는 반복문을 이용해 모든 노드를 가져오고, 이것의 위도/경도 정보를 현재 위치의 경도와 위도와 0.01 이내의 차이를 가진 노드들만 지도에 표시하게 만들어 가까운 노드들만 남기도록 필터링 했다.

지도이(가) 표시된 사진

자동 생성된 설명

<그림> 홍수 발생 시 침수 대피소 안내

이 페이지를 만드는 데 사용된 API는 다른 API들과 달리, 위도&경도로 좌표를 나타낸 것이 아니라, TM128 좌표를 이용해서 위치 정보를 제공했다. 이를 해석하기 위해 3.3.2)에서 소개한 Geocoder를 활용해 위도, 경도 변환한 뒤 행정 구역 이름(예: 서울 특별시)을 얻어 이것을 다시 행정 구역 코드로 변환했다. 이런 과정을 통해 침수에 대처할 수 있는 대피소 위치를 네이버 지도로 제공받을 수 있다.

## 수정&보완할 사항

한 페이지에 네이버 지도 객체가 3개나 있어서 그런지 가끔 지도가 정상적으로 표시되지 않는 상황을 겪었다. 새로 고침 혹은 창의 크기를 조절하면 해결되는 문제지만, 이를 통해서 하나의 지도 객체에 탭 전환 별로 다른 정보를 표시하는 것이 안정성 면에서 좋은 선택이 될 것 같다.

## 의의

이번 과제를 수행하면서 OPEN API의 데이터를 활용하는 다양한 방법들을 알 수 있었다. 데이터가 XML인가, JSON인가 혹은 Text인가 등에 따라 다른 메소드를 이용해야 했다. 이번에는 공공 데이터 포털의 재난 관련 데이터만 이용했지만, 다른 OPEN API를 제공하는 경우 그에 맞는 프로그램 혹은 웹 페이지를 제작하는데 도움이 될 것이다. 피난 도우미 페이지는 정식으로 서비스되지 않아 누군가의 실제 재난 상황에 도움이 될지는 모르겠지만 Ajax를 통한 서버와 클라이언트 사이의 데이터 교환 방식과 HTML, CSS, JS와 친숙해질 수 있었다.

# 결과

## 부록

1. 참고자료(reference)

[1] 한국국토정보공사, *침수흔적관리 시스템 & 대피소 서비스*, 공공 데이터 포털, 2021.11.05수정, 2021.12.14 방문

(url=<https://www.data.go.kr/tcs/dss/selectApiDataDetailView.do?publicDataPk=15058074>)

[2] 행정안전부, *지진 옥외대피장소,* 공공 데이터 포털,2021.3.31 수정, 2021.12.14 방문

(url= <https://www.data.go.kr/tcs/dss/selectApiDataDetailView.do?publicDataPk=15039250>)

[3] 행정안전부, *재난문자방송 발령현황(지역별)*, 2021.10.21 수정, 2021.12.14 방문

(url= <https://www.data.go.kr/tcs/dss/selectApiDataDetailView.do?publicDataPk=15091495>)

[4] 행정안전부, *행정표준코드관리시스템*, 2021.12.14 방문

(url = <https://www.code.go.kr/stdcode/regCodeL.do>)

[5] 네이버, 좌표계 변환하기, 2021.12.14 방문

(url [= https://navermaps.github.io/maps.js.ncp/docs/tutorial-1-geocoder-transcoord.example.html](file:///C:\Users\USER\Downloads\=%20https:\navermaps.github.io\maps.js.ncp\docs\tutorial-1-geocoder-transcoord.example.html))

1. 소스 코드
2. <!DOCTYPE html>
3. <html>
4. <head>
5. <meta charset="UTF-8">
6. <meta http-equiv="X-UA-Compatible" content="IE=edge">
7. <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-scalable=no">
8. <title>피난 도우미</title>
9. <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
10. <script type="text/javascript" src="https://openapi.map.naver.com/openapi/v3/maps.js?ncpClientId=6zgpjicay9"></script>
11. <script type="text/javascript" src="https://openapi.map.naver.com/openapi/v3/maps.js?ncpClientId=6zgpjicay9&submodules=geocoder"></script>
13. <style>
14. h1{
15. background-color: #d9ffd2;
16. font-weight: bold;
17. text-align: center;
19. }
20. span{
21. color:blue;
22. font-weight: bold;
23. text-align: center;
24. }
25. p{
26. text-align: center;
27. }
28. .id{
29. text-align:right;
30. }
31. table,th,td {
32. border : 1px solid #d9ffd2;
33. border-collapse: collapse;
34. }
35. th,td {
36. padding: 10px;
37. }
38. .tab{float:left; width:100%; height:900px;position:relative;}
39. .tabnav{font-size:0; width:600px; border:1px solid #ddd;}
40. .tabnav li{display:inline-block;  height:46px; text-align:center; border-right:1px solid #ddd;}
41. .tabnav li a:before{content:""; position:absolute; left:0; top:0px; width:100%; height:3px; }
42. .tabnav li a.active:before{background:#FFFFFF;}
43. .tabnav li a.active{border-bottom:1px solid #fff;}
44. .tabnav li a{ position:relative; display:block; background: #f8f8f8; color: #000; padding:0 30px; line-height:46px; text-decoration:none; font-size:16px;}
45. .tabnav li a:hover,
46. .tabnav li a.active{background:#d9ffd2; color:rgb(58, 58, 58); }
47. .tabcontent{ text-align:center; padding: 20px; height:100%; border:1px solid #ddd; border-top:none;}
48. </style>
49. </head>
50. <body>
51. <h1>피난 도우미 페이지 </h1> <p>지도가 잘 표시되지 않으면 <span>새로고침</span>을 시도하거나, <span>F12키를 두 번</span> 눌러주세요.</p><p class="id">2018440102 이승원 <br>Email : kkod3327@uos.ac.kr     </p>
52. <div class="tab">
53. <ul class="tabnav">
54. <li><a href="#tab01">재난문자발송현황</a></li>
55. <li><a href="#tab02">지진</a></li>
56. <li><a href="#tab03">홍수</a></li>
57. </ul>
58. <div class="tabcontent">
59. <div id="tab01">
60. <h3>조회하고자 하는 지역을 클릭해주세요.</h3>
61. <div id="map" style="display:inline-block ; width:80%;height:600px;"></div>
62. <table id="내용" style="display:inline-block; width:80%;"></table></div>


66. <div id="tab02">
67. <h3>현재 위치에서 가까운 옥외 대피 장소를 검색합니다.</h3>
68. <div id="map2" style="display:inline-block; width:80%; height:600px;"></div>
69. <table id="내용2" style="display:inline-block; width:80%;"></table>
70. <p id="내용2-2">로딩 중</p>
71. </div>
72. <div id="tab03">
73. <h3>현재 위치에서 가까운 대피 장소를 검색합니다.</h3>
74. <div id="map3" style="display:inline-block; width:80%; height:600px;"></div>
75. <div id="내용3">로딩 중</div>
76. </div>
77. </div>


81. </div><!--tab-->
83. </div>
84. <script>
85. $(function(){
86. $('.tabcontent > div').hide();
87. $('.tabnav a').click(function () {
88. $('.tabcontent > div').hide().filter(this.hash).fadeIn();
89. $('.tabnav a').removeClass('active');
90. $(this).addClass('active');
91. return false;
92. }).filter(':eq(0)').click();
93. });
94. //탭전환
95. let SIDO;
96. //현재 위치 기반
97. navigator.geolocation.getCurrentPosition(function(pos) {
98. 위도 = pos.coords.latitude;
99. 경도 = pos.coords.longitude;
100. var map2 = new naver.maps.Map(document.getElementById('map2'), {
101. zoom: 17,
102. mapTypeId: 'normal',
103. center: new naver.maps.LatLng(위도,경도)
104. });
105. naver.maps.Event.once(map2, 'init\_stylemap', CreateMAP(map2));
106. var map3 = new naver.maps.Map(document.getElementById('map3'), {
107. zoom: 9,
108. mapTypeId: 'normal',
109. center: new naver.maps.LatLng(위도,경도)
110. });
111. let marker = new naver.maps.Marker({
112. position: new naver.maps.LatLng(위도,경도),
113. map: map2,
114. title:"현재 위치"
115. });
116. let markerX = new naver.maps.Marker({
117. position: new naver.maps.LatLng(위도,경도),
118. map: map3,
119. title:"현재 위치"
120. });
121. document.getElementById('내용2').innerHTML+="<br> 현재 위치 위도: "+위도+", 경도: "+경도;
122. naver.maps.Service.reverseGeocode({
123. coords: new naver.maps.LatLng(위도,경도),
124. }, function(status, response) {
125. if (status !== naver.maps.Service.Status.OK) {
126. return alert('Something wrong!');
127. }
128. var result = response.v2, // 검색 결과의 컨테이너
129. items = result.results, // 검색 결과의 배열
130. address = result.address; // 검색 결과로 만든 주소
132. SIDO=items[0].code.id;
133. naver.maps.Event.once(map3, 'init\_stylemap', CreateMAP2(map3,SIDO));
134. });

137. });
138. var HOME\_PATH = "https://navermaps.github.io/maps.js/docs/",
139. urlPrefix = HOME\_PATH +'/data/region',
140. urlSuffix = '.json',
141. regionGeoJson = [],
142. loadCount = 0;
143. var map = new naver.maps.Map(document.getElementById('map'), {
144. zoom: 7,
145. mapTypeId: 'normal',
146. center: new naver.maps.LatLng(36.4203004, 128.317960)
147. });
148. //재난현황 지도생성
149. //지도가 로드 된 이후 단 한번만 수행되는 이벤트
150. naver.maps.Event.once(map, 'init\_stylemap', function () {
151. //총 18개의 행정구역
152. for (var i = 1; i < 18; i++) {
153. var keyword = i +'';
154. if (keyword.length === 1) {
155. keyword = '0'+ keyword;
156. }
157. //키워드 01,02,03 등 생성
158. $.ajax({
159. url: urlPrefix + keyword + urlSuffix,
160. success: function(idx) {            //데이터를 받아오는 것을 success하면 실행
161. return function(geojson) {          //geojson 17개를 받아오면 지도 위에 그림
162. regionGeoJson[idx] = geojson;
163. loadCount++;
164. if (loadCount === 17) {
165. startDataLayer(); //지도 위에 그리는 함수인듯
166. }
167. }
168. }(i - 1)
169. });// GEOJSON 요청
170. }
172. });
174. 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>');
175. //마우스를 올렸을 때 대전광역시, 서울 특별시 같이 이름 표시
176. tooltip.appendTo(map.getPanes().floatPane);
177. function startDataLayer() {
178. map.data.setStyle(function(feature) { //데이터 내부에 일괄적으로 적용되는 스타일
179. var styleOptions = {
180. fillColor: '#ff0000',
181. fillOpacity: 0.0001,
182. strokeColor: '#ff0000',
183. strokeWeight: 2,
184. strokeOpacity: 0.4
185. };
186. //마우스 올렸을 때
187. if (feature.getProperty('focus')) {
188. styleOptions.fillOpacity = 0.6;
189. styleOptions.fillColor = '#0f0';
190. styleOptions.strokeColor = '#0f0';
191. styleOptions.strokeWeight = 4;
192. styleOptions.strokeOpacity = 1;
193. }
195. return styleOptions;
196. });
197. //받아온 GEOJSON 파일을 맵 데이터 객체에 등록
198. regionGeoJson.forEach(function(geojson) {
199. map.data.addGeoJson(geojson);
200. });
201. //마우스 클릭시
202. map.data.addListener('click', function(e) {
203. var feature = e.feature;
204. if (feature.getProperty('focus') !== true) {
205. feature.setProperty('focus', true);
206. CreateTable(feature.getProperty('area1'));
207. } else {
208. feature.setProperty('focus', false);
209. DeleteTable();
210. }
211. });
212. //마우스 올렸을 때
213. map.data.addListener('mouseover', function(e) {
214. //이름 받아오기
215. var feature = e.feature,
216. regionName = feature.getProperty('area1');

219. //개발자 콘솔에서 map.data.feature을 입력해서 확인할 수 있음
220. tooltip.css({
221. display: '',
222. left: e.offset.x,
223. top: e.offset.y
224. }).text(regionName);
225. map.data.overrideStyle(feature, {
226. fillOpacity: 0.6,
227. strokeWeight: 4,
228. strokeOpacity: 1
229. });
230. });
231. //마우스가 벗어나면 기존 툴팁 제거
232. map.data.addListener('mouseout', function(e) {
233. tooltip.hide().empty();
234. map.data.revertStyle();
235. });
236. }
237. function CreateTable(Local\_Name){
238. var xhr = new XMLHttpRequest();
239. var url = 'http://apis.data.go.kr/1741000/DisasterMsg4/getDisasterMsg2List'; /\*URL\*/
240. var queryParams = '?' + encodeURIComponent('serviceKey') + '='+'XdLb2GVZJthcux3Dx0%2BTXO12r6mgf2PYbcBiN6NmH6yrs3fpH0oaji%2B2qHtTfzH%2FbALKJPrNiF11drPPYP657A%3D%3D'; /\*Service Key\*/
241. queryParams += '&' + encodeURIComponent('pageNo') + '=' + encodeURIComponent('1'); /\*\*/
242. queryParams += '&' + encodeURIComponent('numOfRows') + '=' + encodeURIComponent('10'); /\*\*/
243. queryParams += '&' + encodeURIComponent('type') + '=' + encodeURIComponent('xml'); /\*\*/
244. queryParams += '&' + encodeURIComponent('create\_date') + '=' + encodeURIComponent('2021/06/22 00:00:00'); /\*\*/
245. queryParams += '&' + encodeURIComponent('location\_name') + '=' + encodeURIComponent(Local\_Name); /\*\*/
246. xhr.open('GET', url + queryParams);
248. xhr.onreadystatechange = function () {
249. if (this.readyState == 4) {
250. const x = xhr.responseXML;
251. const loc = x.getElementsByTagName("location\_name");
252. const msg = x.getElementsByTagName("msg");
253. let table="<tr><th>지역</th><th>내용</th></tr>";
254. for(let i=0;i<loc.length;i++)
255. {
256. table +="<tr><td>"+
257. loc[i].firstChild.data +
258. "</td><td>" +
259. msg[i].firstChild.data +
260. "</td></tr>";
261. }
262. document.getElementById('내용').innerHTML=table;
263. }
264. }
266. xhr.send('');
267. };
268. function DeleteTable(){
269. document.getElementById("내용").innerHTML="";
270. }
272. function CreateMAP(obj){
273. xhr2 = new Array(10);
274. x2 = new Array(10);
275. for(j=1;j<11;j++){
276. xhr2[j] = new XMLHttpRequest();
277. var url2 = 'http://apis.data.go.kr/1741000/EmergencyAssemblyArea\_Earthquake2/getArea1List'; /\*URL\*/
278. var queryParams2= '?' + encodeURIComponent('serviceKey') + '='+'XdLb2GVZJthcux3Dx0%2BTXO12r6mgf2PYbcBiN6NmH6yrs3fpH0oaji%2B2qHtTfzH%2FbALKJPrNiF11drPPYP657A%3D%3D'; /\*Service Key\*/
279. queryParams2 += '&' + encodeURIComponent('pageNo') + '=' + encodeURIComponent(j); /\*\*/
280. queryParams2 += '&' + encodeURIComponent('numOfRows') + '=' + encodeURIComponent('1000'); /\*\*/
281. queryParams2 += '&' + encodeURIComponent('type') + '=' + encodeURIComponent('xml'); /\*\*/
282. xhr2[j].open('GET', url2 + queryParams2);
283. xhr2[j].send('');
284. xhr2[j].onreadystatechange = function () {
285. if (this.readyState == 4) {
286. x2[j] = this.responseXML;
287. var wuido=x2[j].getElementsByTagName('ycord');
288. var gyungdo=x2[j].getElementsByTagName('xcord');
289. var eqname=x2[j].getElementsByTagName('vt\_acmdfclty\_nm');
291. for(let i=0;i<eqname.length;i++)
292. {
293. if(Math.abs(위도-wuido[i].firstChild.data)<0.01 && Math.abs(경도-gyungdo[i].firstChild.data)<0.01)
294. {
295. let marker= new naver.maps.Marker({
296. position: new naver.maps.LatLng(wuido[i].firstChild.data, gyungdo[i].firstChild.data),
297. map:obj,
298. title:eqname[i].firstChild.data});
299. }
300. else{
301. };
302. document.getElementById("내용2-2").innerHTML="<br>로딩 완료<br> 가까운 옥외 대피소로 신속히 이동하세요.";
303. };
304. };
305. }}};
306. function CreateMAP2(obj2,city\_code){
307. var xhr3 = new XMLHttpRequest();
308. var url3 = 'http://apis.data.go.kr/B553501/flRescueSpService/getFlRescueSpInfo'; /\*URL\*/
309. var queryParams3 = '?' + encodeURIComponent('serviceKey') + '='+'XdLb2GVZJthcux3Dx0%2BTXO12r6mgf2PYbcBiN6NmH6yrs3fpH0oaji%2B2qHtTfzH%2FbALKJPrNiF11drPPYP657A%3D%3D'; /\*Service Key\*/
310. queryParams3 += '&' + encodeURIComponent('pageNo') + '=' + encodeURIComponent('1'); /\*\*/
311. queryParams3 += '&' + encodeURIComponent('numOfRows') + '=' + encodeURIComponent('10'); /\*\*/
312. queryParams3 += '&' + encodeURIComponent('adm\_cd') + '=' + encodeURIComponent(city\_code); /\*\*/
313. xhr3.open('GET', url3 + queryParams3);
314. xhr3.send('');
315. xhr3.onreadystatechange = function () {
316. if (this.readyState == 4) {
318. var x3=xhr3.responseText;
319. var x3=JSON.parse(x3);
320. for(let i=0;i<10;i++)
321. {
322. var wuido3=x3.body.items[i].item.y;
323. var gyungdo3=x3.body.items[i].item.x;
324. var eqname3=x3.body.items[i].item.shelter\_nm;
325. var tm128 =new naver.maps.Point(wuido3,gyungdo3);
326. var jp = naver.maps.TransCoord.fromTM128ToLatLng(tm128);
327. let Kmarker= new naver.maps.Marker({
328. position: new naver.maps.LatLng(jp.y+3.1392125, jp.x-1.4683152),
329. map:obj2,
330. title:eqname3
331. });
332. console.log(jp);
333. };
335. document.getElementById("내용3").innerHTML="<br>로딩 완료<br>신속히 대피소로 이동하세요.";
336. };
338. }};
340. </script>
341. </body>
342. </html>