실습 코드 분석

2021년 12월 14일

2021년 2학기 소프트웨어시스템실습

담당교수: 최성종

이름 : 하 지 호

gkwlgh9595@uos.ac.kr

서울시립대학교 전자전기컴퓨터공학부

차 례

[1. 분석 코드 1](#_Toc89181097)

[2. 주요 개념 정리 2](#_Toc89181100)

[2.1. JSON 2](#_Toc89181101)

[2.2. jQuery-AJAX 2](#_Toc89181102)

[2.3. GeoJSON 2](#_Toc89181103)

[3. 코드에서 사용한 클라스 3](#_Toc89181104)

[4. 실습 코드 분석 4](#_Toc89181108)

[4.1. 클라스의 주된 목적 4](#_Toc89181109)

5. 토론

6.결론

7.부록

1. 분석코드
2. 주요개념정리
   1. Jason
3. Json은 객체의 프로퍼티를 보관 및 전송을 편하게 하기위해서 특정한 규칙으로 스트링화한 데이터이다.

person = {name:"John", age:31, city:"New York"};

1. 위와 같은 객체를 json으로 표현하면 '{"name":"John", "age":30, "car":null}' 이다.

객체 숫자 배열 등을 모두 json으로 처리할 수 있다.

{  
"employee":{"name":"John", "age":30, "city":"New York"}  
}

1. JSON.parse를 이용해 객체화시킬 수 있다.

const obj = JSON.parse('{"name":"John", "age":30, "city":"New York"}');

1. JSON.stringify 를 통해서 다시 스트링화 할 수 있다.

myJSON = JSON.stringify(obj)

* 1. jQuery ajax의 개념

1. ajax는 html파일의 비동기적 요청을 위해서 사용된다.

그리고 dom 객체에 getelementby~의 메소드 없이 selector 와 load 메소드를 이용해서 간단하게 html의 내용을 변경할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. figure2 와 같이 $("#div1").load("demo\_test.txt");를 이용해 div1 id를 가진 html객체의 내용을 변경할 수 있다.
   1. Geojson의 개념

[GeoJSON](http://geojson.org/geojson-spec.html) 객체는 지리 공간 데이터를 기반으로 지도 위에 점, 선, 도형 등을 표현하는 데이터 구조를 정의합니다.

1. 코드에서 사용한 클라스

3.1 naver.maps.Map

Map 클래스는 애플리케이션에서 지도 인스턴스를 정의합니다. 이 객체를 생성함으로써 개발자는 지정한 DOM 요소에 지도를 삽입할 수 있습니다.

naver.maps.Event

이벤트리스너를 정의한다.

Naver.maps.data

Data 클래스는 [GeoJSON](https://navermaps.github.io/maps.js.ncp/docs/global.html" \l "GeoJSON), [KML](https://navermaps.github.io/maps.js.ncp/docs/global.html#KML), [GPX](https://navermaps.github.io/maps.js.ncp/docs/global.html#GPX) 지리 공간 데이터를 기반으로 지도 위에 점, 선, 도형 등을 표현하는 데이터 레이어를 정의합니다.

naver.maps. Feature

Feature 클래스는 FeatureCollection 타입의 형상 정보 모음에서 Feature 타입의 개별 GeoJSON 객체를 추상화합니다.

Method로 foreach, getbounds(), getproperty등이있다.

3.2 코드 분석 (GeoJSON 데이터 레이어 표시하기)

새로운 map이라는 객체를 생성한다.

텍스트, 화면, 스크린샷, 설정이(가) 표시된 사진

자동 생성된 설명

geojson객체를 받아온다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Ajax를 이용해 네이버서버의 특정 ulr로 요청하고 이후에 success 하면 geojson 데이터를 startDataLayer 함수의 첫번째 인수로 넣고startDataLayer함수를 실행한다.

geojson객체의 정보를 바탕으로 스타일 지정하기

텍스트이(가) 표시된 사진

자동 생성된 설명

map이라는 namespace의 data라는 객체의 여러 메소드들을 이용해서 수행한다.

setStyle은 초기의 형상정보를 지정한다.

feature객체는 형상정보를 담고있는 객체를 의미하고 객체내부의 메소드 getproperty를 이용해서 property들을 value를 스트링 형태로 반환 받아볼 수 있다.

Filecolor 프로퍼티는 내부의색

Strokecolor 프로퍼티는 외각선의 색

등등 프로퍼티를 지정하고 리턴하면 그 값을 setStyle의 인수로 받아온다.

이벤트 발생시 스타일 재설정

텍스트이(가) 표시된 사진

자동 생성된 설명

addListener로 해당 이벤트가 발생시 비동기적으로 콜벡함수를 실행시킨다.

Feature 객체의 getbounds를 이용해서 객체좌표 경계값에 해당하는 프로퍼티를 반환받을수 있다.

Map.data 객체의 여러 매소드를 이용해서 geojson의 스타일을 수정할 수 있다.

매소드에 대한 설명은 위 코드의 주석으로 설명하겠음.

1. 실습코드분석

공공데이터포털 api호출

 function loadDoc(locationnumber,startdate, lastdate) {

//startdate, lastdate에서 ‘-’기호를 제거해주고 20211210같은 날짜형식으로 변경

            startdate = startdate.replace("-","");

            startdate = startdate.replace("-","");

            lastdate = lastdate.replace("-","");

            lastdate = lastdate.replace("-","");

          document.getElementById('location').innerText = locationnumber;

var url = 'http://openapi.data.go.kr/openapi/service/rest/Covid19/getCovid19SidoInfStateJson'; */\*URL\*/*

var queryParams = '?' + encodeURIComponent('serviceKey') + '='+'xdsJjk0%2FgxijrqmY1DqXfmf3UZ%2Fum3TRvropo81dSvBAyqWLhzF01ARJK0C6BExA3QXkoJj2iIHDk2L%2BkAFqQg%3D%3D'; */\*Service Key\*/*

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

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

queryParams += '&' + encodeURIComponent('startCreateDt') + '=' + encodeURIComponent(startdate); */\*\*/*

queryParams += '&' + encodeURIComponent('endCreateDt') + '=' + encodeURIComponent(lastdate); */\*\*/*

          const xhttp = new XMLHttpRequest();

          xhttp.onload = function() {myFunction(this, locationnumber);}

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

          xhttp.send();

        }

Xml 파일을 가공하고 표와 그래프 생성

function myFunction(xml, locationnumber) {

  const xmlDoc = xml.responseXML;

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

  let incDecdata=new Array();

  let date=new Array();

  let j=0;

  let sum=0;

  let table="<th>해당기간 총 확진자수</th>"; //총확진자를 구하기위한 표의 일부

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

*if*(x[i].getElementsByTagName("gubun")[0].childNodes[0].nodeValue==locationnumber)

      {

    let t = 1\*x[i].getElementsByTagName("incDec")[0].childNodes[0].nodeValue

//확진자수 문자열을 숫자로 변경

    sum += t; //확진자수의 기간별 총합을 구하기 위해 더한다.

     }

*if*(x[i].getElementsByTagName("gubun")[0].childNodes[0].nodeValue==locationnumber)

      { // 지역의 이름과 일치하는 확진자수와 날짜를 구해서 incDecdata와 date에 배열로 저장한다.

    incDecdata[j] = x[i].getElementsByTagName("incDec")[0].childNodes[0].nodeValue;

    date[j] = x[i].getElementsByTagName("stdDay")[0].childNodes[0].nodeValue;

    j++;

     }

  }

  var a={ // 그래프에 관한 정보를 담은 객체

    bindto: "#linechart",

    data: { //column은 그래프의 y축 변수를 의미하고 지금은 비어있다.

        columns: [

      ]

    },

    axis: {

        x: { // x축값이며 현재 비어있다.

            type: 'category',

            categories:[]

        }},

        options: {

      responsive: false, // 그래프의 크기가 정적으로 변화한다.

      scales: {

        yAxes: [{

          ticks: {

            beginAtZero: true

          }

        }]

      },

    }

  };

  xname=locationnumber+'일일확진자수'; // 범례에들어갈 이름( OO일일확진자수)

  incDecdata.unshift("'"+xname+"'"); //확진자수 배열의 첫번째값은 범례이다.

  a.axis.x.categories=date; //x축에 들어가는 값은 날짜

  a.data.columns=[ incDecdata]; //y축에들어갈 확진자수

  var chart = c3.generate(a); //그래프생성

  table += "<tr><td>" +

    sum+'명'

    "</td></tr>"; //sum은 기간내의 총 확진자수

  document.getElementById("demo").innerHTML = table; //표생성

}

Html

텍스트이(가) 표시된 사진

자동 생성된 설명

  <form>//위의 사진과 같은 구성을 만드는 html소스이다.

            <label>우리동네이름은?</label>

            <select id="input\_loc">

              <option>서울</option>

              <option>부산</option>

              <option>대구</option>

              <option>인천</option>

              <option>광주</option>

              <option>대전</option>

              <option>울산</option>

              <option>세종</option>

              <option>경기</option>

              <option>강원</option>

              <option>충북</option>

              <option>충남</option>

              <option>전북</option>

              <option>전남</option>

              <option>경북</option>

              <option>경남</option>

              <option>제주</option>

            </select>

        </form>

      <form>

        <p><input id="startdate" type="date" value="2021-12-01" min="2020-04-10" max="2021-12-13">부터</p>

      </form>

      <form>

        <p><input id="lastdate" type="date" value="2021-12-13" min="2020-04-10" max="2021-12-13">까지</p>

      </form>

      <p><span id="location">OO</span> 지역의 확진자수입니다.</p>

      <table id="demo"></table>

<button type="button" onclick="loadDoc(document.getElementById('input\_loc').options[document.getElementById('input\_loc').selectedIndex].value,document.getElementById('startdate').value,document.getElementById('lastdate').value);">입력</button>//입력버튼을 클릭하면 loaddoc이라는함수와 myfunction이라는 함수가 순차적으로 실행된다.

<br><br>

<div id="linechart" style="width:800px;height:300px;"></div> //차트

Navermap

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

            var feature = e.feature;

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

*for* (i = 0; i < map.data.\_features.length; i++) {

                map.data.\_features[i].setProperty('focus', false);

                }

                map.data.revertStyle();

                let locationname;

*switch* (feature.getProperty('area1')){

//feature에 클릭한 지역의 정보가 담긴다. 이것을 공공데이터 포털의 지역이름과 호환시켜주기위해 스위치문을 사용했다.

*case*  "서울특별시" :

    locationname = "서울";

*break*;

*case*  "부산광역시" :

    locationname = "부산";

*break*;

*case* "대구광역시":

    locationname = "대구";

*break*;

*case* '인천광역시':

    locationname = "인천";

*break*;

*case* "광주광역시":

    locationname = "광주";

*break*;

*case* "대전광역시":

    locationname = "대전";

*break*;

*case* "울산광역시":

    locationname = "울산";

*break*;

*case* "세종특별자치시":

    locationname = "세종"

*break*;

*case* "경기도":

    locationname = "경기";

*break*;

*case* "강원도":

    locationname = "강원";

*break*;

*case* "충청북도":

    locationname = "충북";

*break*;

*case* "충청남도":

    locationname = "충남";

*break*;

*case* "전라북도":

    locationname = "전북";

*break*;

*case* "전라남도":

    locationname = "전남";

*break*;

*case* "경상북도":

    locationname = "경북";

*break*;

*case* "경상남도":

    locationname = "경남";

*break*;

*case* "제주특별자치도":

    locationname = "제주";

*break*;

  }

*// earthquake\_load(feature.getProperty('area1'));*

               loadDoc(locationname,//공공데이터 포털에서 xml파일을 호출하는 함수

               document.getElementById('startdate').value,document.getElementById('lastdate').value);

                feature.setProperty('focus', true);

            } *else* {

                feature.setProperty('focus', false);

            }

            });

1. 토론
   1. 요구사항

1) form 으로 지역과 날짜정보를 입력

2) 지도를 클릭해서 원하는 지역을 입력.

3) 해당지역의 기간별 확진자 수를 그래프로 출력

4) 해당지역의 총 확진자수 표로 출력

위 기능을 모두다 수행하고 있기 때문에 기능적 요구사항은 모두 만족한다.

5.2 고쳐야할 점 :

그래프의 배치를 지도옆에 하면 깔끔하고 동적으로 지도와 그래프의 크기가 변한다면 더 좋은 유저인터페이스를 가질 수 있을것이다. 이를 위해서 c3.js의 객체의 옵션을 조정하는 방안을 생각할 수 있다.

1. 결론
   * 과제 구현 결과 : 발표동영상의 시연과정 참고
   * 향후 방향 : 사용자의 수가 1000명이상이 되면 공공데이터 포털에서 제공하는 api 횟수를 초과한다. 따라서 데이터베이스에 매일매일 xml파일을 업데이트하고 이를 이용해서 서비스를 제공하면 매일 1000회 이상 사용해도 사이트가 정상적인 서비스를 제공할 수 있을 것이다, 이를 위해서 데이터베이스를 구성하고 이를 csv파일로 만들어 저장하는 방안에 대해서 생각해 볼 필요가 있다.
2. 부록:
   * Navermap, c3.js, 공공데이터포털 api를 이용
   * 전체 소스코드

* <!DOCTYPE html>
* <html lang="ko">
* <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">
* <title>우리동네 확진자</title>
* <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/c3/0.4.11/c3.min.css"/>
* <script src="https://d3js.org/d3.v3.min.js"></script>
* <script src="https://cdnjs.cloudflare.com/ajax/libs/c3/0.4.11/c3.min.js"></script>
* <style>
* th, td {
* border: 1px solid black;
* }
* </style>
* *<!-- 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=3j74nj8hha"></script>
* *<!--지진발생 대피소-->*
* </head>
* <body>
* <h3>우리동네 확진자</h3>
* <hr>
* <div id="map" style="width:500px;height:400px;"></div>
* *<!--표 작성-->*
* <table id="list"></table>
* <style>
* table,th,td {
* border : 1px solid black;
* border-collapse: collapse;
* }
* th,td {
* padding: 5px;
* }
* th, td {
* text-align: center;
* }
* </style>
* <script>
* function loadDoc(locationnumber,startdate, lastdate) {
* startdate = startdate.replace("-","");
* startdate = startdate.replace("-","");
* lastdate = lastdate.replace("-","");
* lastdate = lastdate.replace("-","");
* document.getElementById('location').innerText = locationnumber;
* var url = 'http://openapi.data.go.kr/openapi/service/rest/Covid19/getCovid19SidoInfStateJson'; */\*URL\*/*
* var queryParams = '?' + encodeURIComponent('serviceKey') + '='+'xdsJjk0%2FgxijrqmY1DqXfmf3UZ%2Fum3TRvropo81dSvBAyqWLhzF01ARJK0C6BExA3QXkoJj2iIHDk2L%2BkAFqQg%3D%3D'; */\*Service Key\*/*
* queryParams += '&' + encodeURIComponent('pageNo') + '=' + encodeURIComponent('1'); */\*\*/*
* queryParams += '&' + encodeURIComponent('numOfRows') + '=' + encodeURIComponent('10'); */\*\*/*
* queryParams += '&' + encodeURIComponent('startCreateDt') + '=' + encodeURIComponent(startdate); */\*\*/*
* queryParams += '&' + encodeURIComponent('endCreateDt') + '=' + encodeURIComponent(lastdate); */\*\*/*
* const xhttp = new XMLHttpRequest();
* xhttp.onload = function() {myFunction(this, locationnumber);}
* xhttp.open("GET",  url + queryParams);
* xhttp.send();
* }
* function myFunction(xml, locationnumber) {
* const xmlDoc = xml.responseXML;
* const x = xmlDoc.getElementsByTagName("item");
* let incDecdata=new Array();
* let date=new Array();
* let j=0;
* let sum=0;
* let table="<th>해당기간 총 확진자수</th>";
* *for* (let i = 0; i <x.length;  i++) {
* *if*(x[i].getElementsByTagName("gubun")[0].childNodes[0].nodeValue==locationnumber)
* {
* let t = 1\*x[i].getElementsByTagName("incDec")[0].childNodes[0].nodeValue
* sum += t;
* }
* *if*(x[i].getElementsByTagName("gubun")[0].childNodes[0].nodeValue==locationnumber)
* {
* incDecdata[j] = x[i].getElementsByTagName("incDec")[0].childNodes[0].nodeValue;
* date[j] = x[i].getElementsByTagName("stdDay")[0].childNodes[0].nodeValue;
* j++;
* }
* }
* var a={
* bindto: "#linechart",
* data: {
* columns: [
* ]
* },
* axis: {
* x: {
* type: 'category',
* categories:[]
* }},
* options: {
* responsive: false,
* scales: {
* yAxes: [{
* ticks: {
* beginAtZero: true
* }
* }]
* },
* }
* };
* xname=locationnumber+'일일확진자수';
* incDecdata.unshift("'"+xname+"'");
* a.axis.x.categories=date;
* a.data.columns=[ incDecdata];
* var chart = c3.generate(a);
* table += "<tr><td>" +
* sum+'명'
* "</td></tr>";
* document.getElementById("demo").innerHTML = table;
* }
* </script>
* <form>
* <label>우리동네이름은?</label>
* <select id="input\_loc">
* <option>서울</option>
* <option>부산</option>
* <option>대구</option>
* <option>인천</option>
* <option>광주</option>
* <option>대전</option>
* <option>울산</option>
* <option>세종</option>
* <option>경기</option>
* <option>강원</option>
* <option>충북</option>
* <option>충남</option>
* <option>전북</option>
* <option>전남</option>
* <option>경북</option>
* <option>경남</option>
* <option>제주</option>
* </select>
* </form>
* <form>
* <p><input id="startdate" type="date" value="2021-12-01" min="2020-04-10" max="2021-12-13">부터</p>
* </form>
* <form>
* <p><input id="lastdate" type="date" value="2021-12-13" min="2020-04-10" max="2021-12-13">까지</p>
* </form>
* <p><span id="location">OO</span> 지역의 확진자수입니다.</p>
* <table id="demo"></table>
* <button type="button" onclick="loadDoc(document.getElementById('input\_loc').options[document.getElementById('input\_loc').selectedIndex].value,document.getElementById('startdate').value,document.getElementById('lastdate').value);">입력</button>
* <br><br>
* <div id="linechart" style="width:800px;height:300px;"></div>
* *<!--전국 지도 띄우기-->*
* <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)
* });
* 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
* };
* *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);
* });
* map.data.addListener('click', function (e) {
* var feature = e.feature;
* *if* (feature.getProperty('focus') !== true) {
* *for* (i = 0; i < map.data.\_features.length; i++) {
* map.data.\_features[i].setProperty('focus', false);
* }
* map.data.revertStyle();
* let locationname;
* *switch* (feature.getProperty('area1')){
* *case*  "서울특별시" :
* locationname = "서울";
* *break*;
* *case*  "부산광역시" :
* locationname = "부산";
* *break*;
* *case* "대구광역시":
* locationname = "대구";
* *break*;
* *case* '인천광역시':
* locationname = "인천";
* *break*;
* *case* "광주광역시":
* locationname = "광주";
* *break*;
* *case* "대전광역시":
* locationname = "대전";
* *break*;
* *case* "울산광역시":
* locationname = "울산";
* *break*;
* *case* "세종특별자치시":
* locationname = "세종"
* *break*;
* *case* "경기도":
* locationname = "경기";
* *break*;
* *case* "강원도":
* locationname = "강원";
* *break*;
* *case* "충청북도":
* locationname = "충북";
* *break*;
* *case* "충청남도":
* locationname = "충남";
* *break*;
* *case* "전라북도":
* locationname = "전북";
* *break*;
* *case* "전라남도":
* locationname = "전남";
* *break*;
* *case* "경상북도":
* locationname = "경북";
* *break*;
* *case* "경상남도":
* locationname = "경남";
* *break*;
* *case* "제주특별자치도":
* locationname = "제주";
* *break*;
* }
* *// earthquake\_load(feature.getProperty('area1'));*
* loadDoc(locationname,
* document.getElementById('startdate').value,document.getElementById('lastdate').value);
* feature.setProperty('focus', true);
* } *else* {
* 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();
* });
* }
* </script>
* </body>
* </html>