예제 코드 분석

2021년 11월 30일

2021년 2학기 소프트웨어시스템실습

담당교수: 최성종

오창윤

ogaoga5959@office.uos.ac.kr

서울시립대학교 전자전기컴퓨터공학부

차 례

[1. GeoJSON 데이터 레이어 표시하기 1](#_Toc54003187)

[1.1. 예제코드 1](#_Toc54003188)

[1.2. 코드에서 사용한 클래스 1](#_Toc54003188)

[1.3. 실습 코드 분석 1](#_Toc54003188)

# GeoJSON 데이터 레이어 표시하기

## 예제코드

**var** map = **new** naver.maps.Map(document.getElementById('map'), {

zoom: 18,

mapTypeId: 'normal',

center: **new** naver.maps.LatLng(37.3586524, 127.1060678)

});

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

naver.maps.Event.once(map, 'init\_stylemap', **function** () {

$.ajax({

url: HOME\_PATH +'/data/naver.json',

dataType: 'json',

success: startDataLayer

});

});

**function** **startDataLayer**(geojson) {

map.data.addGeoJson(geojson);

map.data.setStyle(**function**(feature) {

**var** color = 'red';

**if** (feature.getProperty('isColorful')) {

color = feature.getProperty('color');

}

**return** {

fillColor: color,

strokeColor: color,

strokeWeight: 2,

icon: null

};

});

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

e.feature.setProperty('isColorful', true);

});

map.data.addListener('dblclick', **function**(e) {

**var** bounds = e.feature.getBounds();

**if** (bounds) {

map.panToBounds(bounds);

}

});

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

map.data.overrideStyle(e.feature, {

strokeWeight: 8,

icon: HOME\_PATH +'/img/example/pin\_spot.png'

});

});

map.data.addListener('mouseout', **function**(e) {

map.data.revertStyle();

});

}

## 코드에서 사용한 클래스

### 클래스의 주된 목적

클래스(class)란 객체를 정의하는 틀 또는 설계도와 같은 의미로 사용된다. 이러한 설계도인 클래스를 가지고, 여러 객체를 생성하여 사용한다. 클래스는 객체의 상태를 나타내는 필드(field)와 객체의 행동을 나타내는 메소드(method)로 구성된다. 즉, 필드(field)란 클래스에 포함된 변수(variable)를 의미한다. 또한, 메소드(method)란 어떠한 특정 작업을 수행하기 위한 명령문의 집합이다.

이러한 클래스를 통해 설계도를 활용하여 주된 기능을 손쉽게 활용할 수 있다.

### 코드에서 사용한 클래스, 프로퍼티, 메소드 기능분석

**var** map = **new** naver.maps.Map(document.getElementById('map'), {

zoom: 18,

mapTypeId: 'normal',

center: **new** naver.maps.LatLng(37.3586524, 127.1060678)

});

Naver.maps.Map

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

mapTypes : 지도 유형의 컬렉션을 포함하는 객체입니다.

Naver.maps.LatLng

LatLng 클래스는 위/경도 좌표를 정의합니다.

Add, sub, mul, div, round, floor 메서드는 LatLng 클래스에서는 사용할 수 없습니다.

naver.maps.Event.once(map, 'init\_stylemap', **function** () {

$.ajax({

url: HOME\_PATH +'/data/naver.json',

dataType: 'json',

success: startDataLayer

});

**function** **startDataLayer**(geojson) {

map.data.addGeoJson(geojson);

기본적으로 모든 map 객체는 data 객체를 포함합니다. 따라서 map.data.addGeoJson 메서드를 사용합니다.

map.data.setStyle(**function**(feature) {

**var** color = 'red';

**if** (feature.getProperty('isColorful')) {

color = feature.getProperty('color');

}

**return** {

fillColor: color,

strokeColor: color,

strokeWeight: 2,

icon: null

};

GeoJSON 지리 공산 데이터의 형상 정보의 스타일을 지정하려면setstyle  메서드를 사용합니다. 이 메서드는 data 레이어의 모든 Feature 객체에 영향을 미치며, StyleOptions 객체 리터럴 또는 Stylingfunction 함수를 인수로 사용합니다.

### 객체 리터럴로 스타일 지정하기

[StyleOptions](https://navermaps.github.io/maps.js.ncp/docs/naver.maps.Data.html#~StyleOptions) 객체 리터럴에서 설정한 스타일은 모든 [Feature](https://navermaps.github.io/maps.js.ncp/docs/naver.maps.Feature.html) 객체에 적용됩니다. 단, 각 [Feature](https://navermaps.github.io/maps.js.ncp/docs/naver.maps.Feature.html) 객체의 기하 구조(geometry) 타입에 따라 적용되는 범위는 다릅니다. 예를 들어, icon 스타일은 [Marker](https://navermaps.github.io/maps.js.ncp/docs/naver.maps.Marker.html) 오버레이 타입인 Point, MultiPoint 기하 구조에만 영향을 줍니다.

### StylingFunction 함수로 스타일 지정하기

[StylingFunction](https://navermaps.github.io/maps.js.ncp/docs/naver.maps.Data.html#~StylingFunction) 함수를 이용하면 각 형상 정보([Feature](https://navermaps.github.io/maps.js.ncp/docs/naver.maps.Feature.html) 객체)의 속성이나 특성에 따라 다른 스타일을 지정할 수 있습니다.

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

e.feature.setProperty('isColorful', true);

});

map.data.addListener('dblclick', **function**(e) {

**var** bounds = e.feature.getBounds();

**if** (bounds) {

map.panToBounds(bounds);

}

## 지정된 스타일 재정의하기

앞서 설명한대로 setStyle 메서드는 [Data](https://navermaps.github.io/maps.js.ncp/docs/naver.maps.Data.html) 레이어의 모든 [Feature](https://navermaps.github.io/maps.js.ncp/docs/naver.maps.Feature.html) 객체에 영향을 미칩니다(이를 전역 스타일이라고 함). 특정 [Feature](https://navermaps.github.io/maps.js.ncp/docs/naver.maps.Feature.html) 객체에 특별한 스타일을 적용하고자 할 때는 overrideStyle 메서드를 사용합니다. 이 메서드는 전역 스타일에 추가로 적용됩니다.

재정의한 스타일을 이전 스타일로 다시 복원하려면 revertStyle 메서드를 사용합니다. 이 메서드는 특정 [Feature](https://navermaps.github.io/maps.js.ncp/docs/naver.maps.Feature.html) 객체에 사용할 수 있지만, 인수를 생략하면 모든 [Feature](https://navermaps.github.io/maps.js.ncp/docs/naver.maps.Feature.html) 객체의 스타일을 복원합니다.

다음은 마우스 포인터가 기하 구조의 경계에 들어올 때와 벗어날 때 스타일을 변경하고 복원하는 예제입니다.

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

map.data.overrideStyle(e.feature, {

strokeWeight: 8,

icon: HOME\_PATH +'/img/example/pin\_spot.png'

});

});

map.data.addListener('mouseout', **function**(e) {

map.data.revertStyle();

});

}

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

실습 코드 분석

**var** map = **new** naver.maps.Map(document.getElementById('map'), {

zoom: 18,

mapTypeId: 'normal',

center: **new** naver.maps.LatLng(37.3586524, 127.1060678)

});

//map을 참조변수로 가지는 Map 객체 생성

//지도를 객체로 받아 프로퍼티와 메소드 정의

//zoom 프로퍼티는 18, mapTypeId 프로퍼티는 normal, center 프로퍼티는 LatLng

//객체의 (37.3586524, 127.1060678)에 해당하는 객체

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

naver.maps.Event.once(map, 'init\_stylemap', **function** () {

$.ajax({

url: HOME\_PATH +'/data/naver.json',

dataType: 'json',

success: startDataLayer

});

});

//주소를 받는 변수지어

//Event 객체의 once 메소드 호출하여 ajax로 받기

**function** **startDataLayer**(geojson) {

map.data.addGeoJson(geojson);

map.data.setStyle(**function**(feature) {

**var** color = 'red';

**if** (feature.getProperty('isColorful')) {

color = feature.getProperty('color');

}

**return** {

fillColor: color,

strokeColor: color,

strokeWeight: 2,

icon: null

};

});

//완료되면 함수실행, 기본설정지정

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

e.feature.setProperty('isColorful', true);

});

//클릭하면 이벤트함수 실행

//이벤트 함수의 feature setProperty 메소드에 의해서 색 변경

map.data.addListener('dblclick', **function**(e) {

**var** bounds = e.feature.getBounds();

**if** (bounds) {

map.panToBounds(bounds);

}

});

//더블클릭 할 시 이벤트 함수 실행

//이벤트 함수 feature의 getBounds 메소드에 의해 형상 정보의 좌표 경계를 받음

//글자 좌표 경계를 포함하는 위치로 지도를 확대

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

map.data.overrideStyle(e.feature, {

strokeWeight: 8,

icon: HOME\_PATH +'/img/example/pin\_spot.png'

});

});

map.data.addListener('mouseout', **function**(e) {

map.data.revertStyle();

});

}

//마우스오버 할 시, 이벤트 함수 실행

//overrideStyle 메소드로 지정된 곳의 아이콘 변경

//마우스아웃 할 시, 이벤트 함수 실행

//revertStyle 메소드로 지정된 곳의 변경사항 원상복귀