FingerEyes-Xr for HTML5 Tutorials - 09

그리드 레이어를 이용한 밀도 분석

2015년 3월 2일, 1차 배포

㈜지오서비스

Grid 레이어를 이용한 밀도 분석

그리드(Grid) 레이어를 이용한 분석 기능 중 밀도 분석 기능을 FingerEyes-Xr for HTML5에서 어떻게 구현할 수 있는지를 설명합니다. FingerEyes-Xr for HTML5에서 제공하는 밀도 분석은 가중치 값을 고려할 수 있으며 전문적인 GIS 분석 시스템에서 생성할 수 있는 품질과 동일한 결과를 제공합니다.

FingerEyes-Xr은 Flex 버전과 HTML5 버전이 존재하며 이 글은 HTML5에 대해 글입니다. FingerEyes-Xr for HTML5에 대한 소스 코드는 GitHub에서 다운로드 받을 수 있으며 URL은 https://github.com/FingerEyes-Xr/src 입니다. 아래의 화면은 FingerEyes-Xr for HTML5에 대한 GitHub 웹 화면입니다.

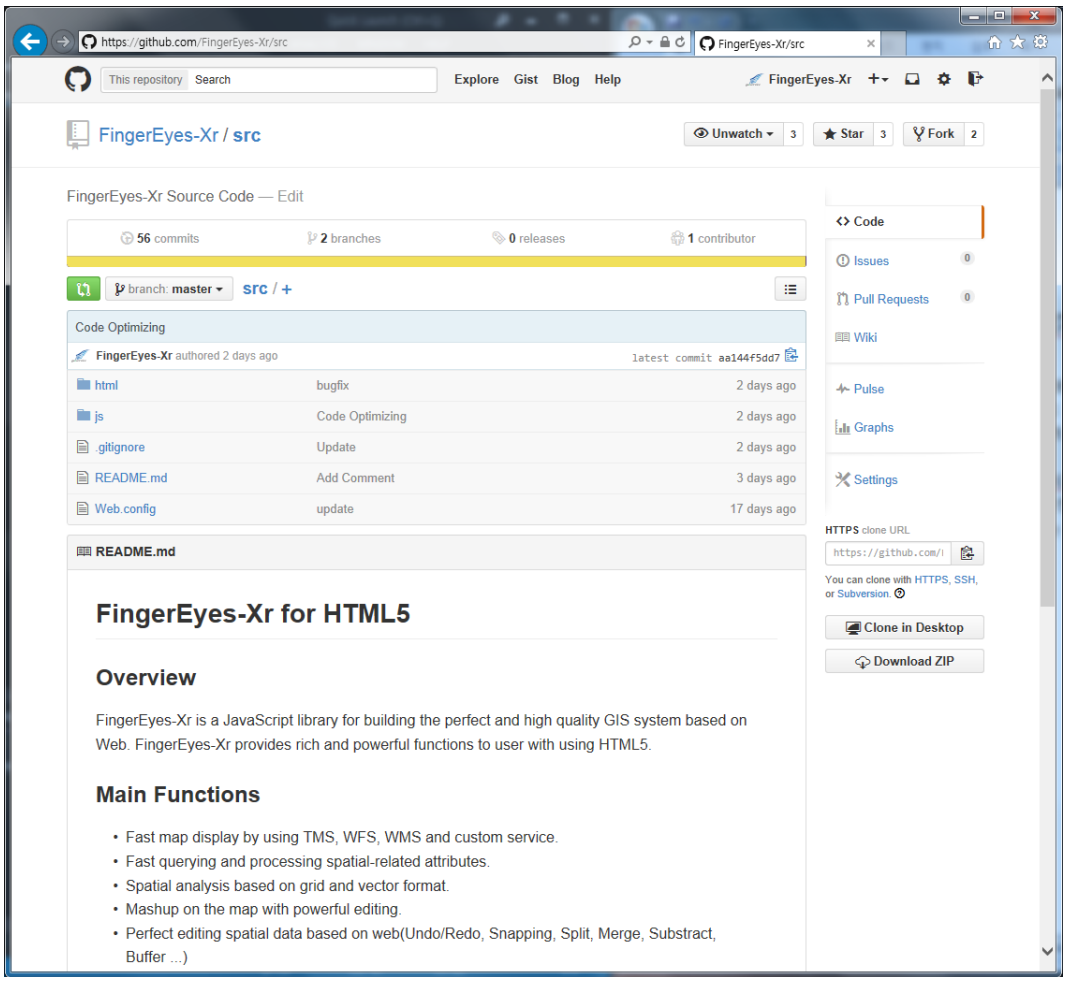

그림 1. FingerEyes-Xr for HTML에 대한 GitHub

필자는 이 글에 대한 예제 코드를 VisualStudio 2012를 이용하여 작성 하였습니다. 독자 여러분들이 어떤 툴을 사용하든 문제는 없겠으나 VisualStudio 2012에서 제공하는 JavaScript 디버깅과 기본적으로 제공하는 웹서버(IIS)를 통해 웹 기반의 프로그래밍을 편리하고 효율적으로 진행할 수 있었기에 선택하였습니다. 참고로 웹 기반의 프로그래밍은 웹서버를 이용하여 URL 형태로 접근하지 않으면 올바르게 실행되지 않으므로 반드시 웹서버를 통해 URL 형태로 접근하시기 바랍니다.

이제 본격적으로 그리드 레이어를 FingerEyes-Xr에서 사용하여 밀도 분석에 대한 코드를 작성해 보겠습니다. 먼저 map.html 파일을 웹서버에서 접근할 수 있는 곳에 생성합니다. 필요하다면 map.html이 아닌 다른 파일명으로 생성 하여도 진행에 문제는 없습니다. 다만, URL을 통해 웹브라우저에서 실행할 수 있는 경로에 생성한다는 것이 중요합니다. 일반 html을 생성하고 다음처럼 입력합니다.

```
01 <html xmlns="http://www.w3.org/1999/xhtml">
02 <head>
03     <title>FingerEyes-Xr</title>
04 </head>
05 <body>
06
07 </body>
08 </html>
```

코드 1. 초기 map.html

위의 코드에서 3번 라인 밑에 다음 코드를 추가합니다.

```
01 <script src="http://www.geoservice.co.kr/z/Xr.min.1.0.js"></script>
02
03 <script type="text/javascript">
04
05 </script>
```

코드 2. 기본 Script

1번 코드에서 CDN(Content Delivery Network)을 사용하여 FingerEyes-Xr의 JavaScript 라이브러리에 대한 소스 코드를 추가 하고 있습니다. 그리고 4번의 빈 공백 부분에 앞으로 추가할 JavaScript 코드가 입력될 것입니다.

UI를 구성하기 위해 <body> 부분을 다음처럼 변경합니다.

```
01 <body>
02     <div id="mainLayout">
03         <div id="mapDiv"></div>
04         <div id="title">
05             FingerEyes-Xr for HTML5 : <font color="#349bd6">GridLayer for Density Example</font>
06             <br />
07             <input type="button" id="density" value="밀도분석실행" onclick="onDensity();">
08             <br />
09             <font size="2" color="#ffffff"><span>가중치값 필드</span>
10             <SELECT id='attributes' size='1' style="width:120px"
11                 onchange="onWeightedValueFieldChange()"></SELECT>
12             <span>그리드해상도</span> <input type="text" style="width:120px" id="resolution">
13             </span>밀도분석반경 </span><input type="text" style="width:120px" id="radius">
14             </font>
15             <div id="progressbar-bg"><div id="progressbar"></div></div>
```

```
16          </div>
17      </div>
18  </body>
```

코드 3. 기본 UI 구성

특히 Id가 mapDiv인 DIV 요소에 맵(Map)이 표시될 것입니다. 일단 여기까지 작성하고 실행해 보면 다음과 같은 화면이 나타나는 것을 볼 수 있습니다.

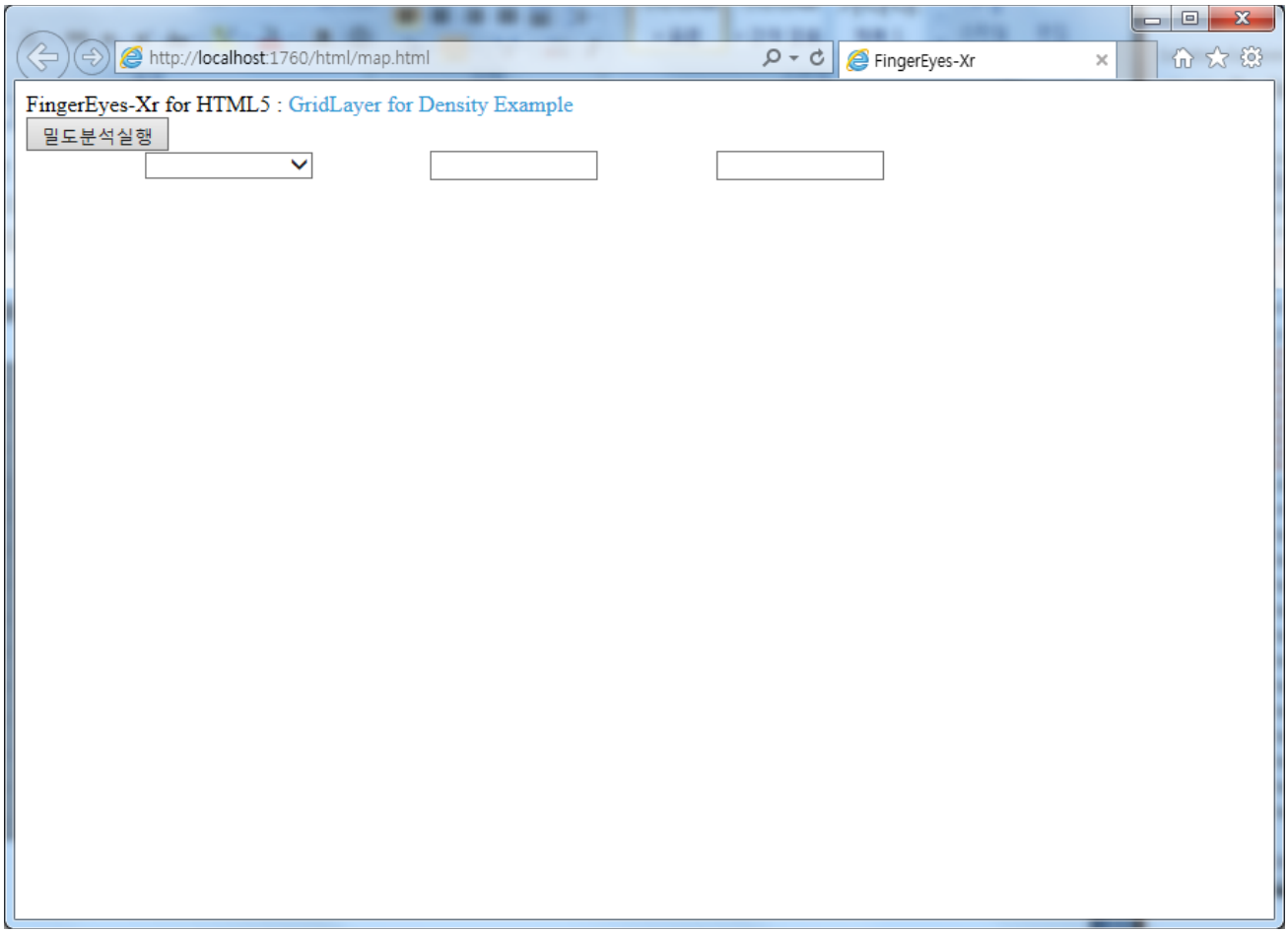

그림 2. 기본 UI 구성에 대한 실행 화면

CSS를 이용한 스타일(Style)이 적용되지 않았는데, 페이지에 스타일을 적용해 보도록 하겠습니다. <head> 바로 밑에 아래의 스타일 코드를 추가합니다.

```
01  <style>
02      body
03      {
04          margin:0px;
05          padding:0px;
06      }
07
08      #mainLayout
09      {
10          width:100%;
11          height:100%;
```

```
12          border:none;
13      }
14
15      #mapDiv
16      {
17          top:0px;
18          left:0px;
19          position:relative;
20          width:100%;
21          height:100%;
22          border:none;
23          overflow:hidden;
24      }
25
26      #title
27      {
28          top:12px;
29          left:12px;
30          padding: 12px;
31          position:absolute;
32          background:rgba(0,0,0,0.7);
33          border:none;
34          overflow:auto;
35          border-radius: 12px;
36          font-size: 24px;
37          color: #ffffff;
38          font-family: "맑은 고딕";
39      }
40
41      #progressbar-bg
42      {
43          width: 510px;
44          background: gray;
45      }
46
47      #progressbar
48      {
49          height: 16px;
50          margin: 1px 0px;
51          color: #fff;
52          padding: 5px;
53          background: rgb(0,0,0);
54          font-size: 14px;
55      }
56  </style>
```

코드 4. UI에 대한 스타일 적용

스타일이 적용된 페이지는 다음과 같습니다.

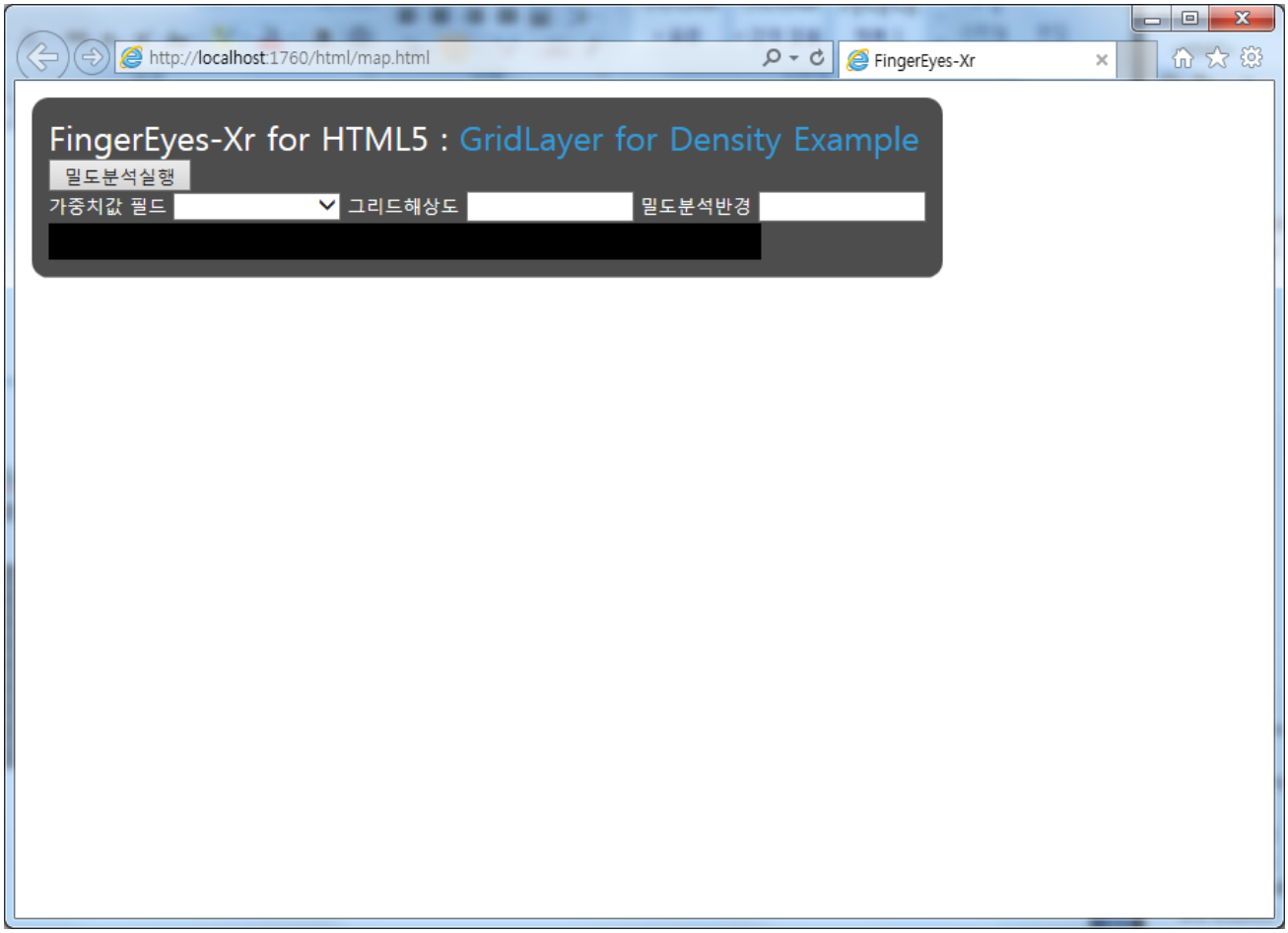

그림 3. 스타일이 적용된 UI

UI에 대해 설명하면 다음 그림과 같습니다.

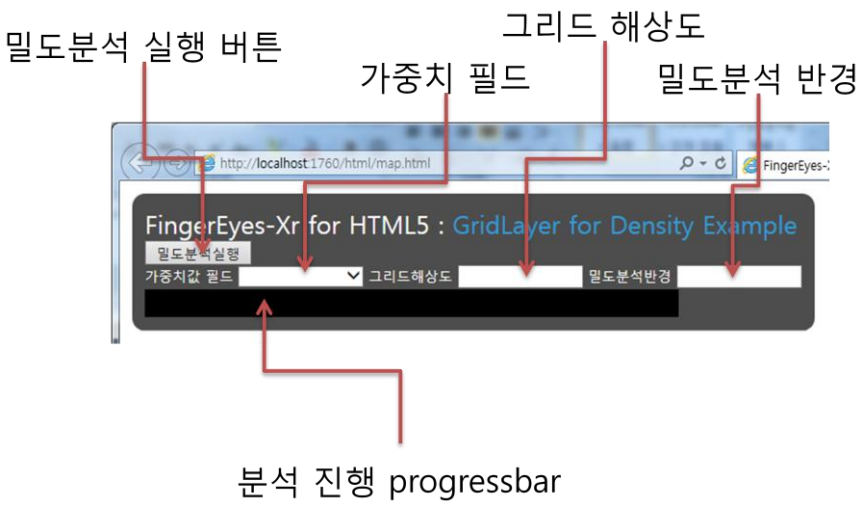

그림 4. UI 설명

밀도 분석 실행 버튼을 누르면 그리드 해상도를 이용해 그리드 레이어를 생성하고 가중치 필드에 대한 값과 밀도 분석 반경 값을 이용해 밀도 분석을 진행합니다. 밀도 분석에 대한 진행율은 분석 진행

progressbar를 통해 사용자에게 피드백 됩니다.

이제 UI와 스타일 적용이 완료되었으므로, 이제 코드를 작성해 보겠습니다. **코드 2. 기본 Script**에서 4번 줄에 다음 코드를 추가합니다.

```
01  var map = null;
```

코드 5. DIV와 연결될 지도 객체 정의

이 map 변수는 지도 객체에 대한 참조로 사용됩니다. 이제 <body>에 대한 onload 이벤트를 load() 함수로 지정합니다.

```
01  <body onload="load()">
```

코드 6. Body 요소의 onload 이벤트 지정

그리고 load 함수를 다음처럼 추가합니다.

```
01  function load() {
02      map = new Xr.Map("mapDiv", {});
03
04
05
06  }
```

코드 7. Body의 onload 이벤트 함수

2번 코드는 id가 mapDiv인 DIV를 이용하여 지도 객체를 생성하고 이를 앞서 정의해 둔 map 객체에 저장하고 있습니다.

레이어를 2개 추가하겠습니다. 첫번째는 배경지도로 사용할 레이어이고 두번째는 밀도 분석을 위한 입력 데이터를 제공하는 포인트 형식의 수치지도 레이어입니다. 먼저 배경지도 레이어 추가를 하겠습니다. 아래의 코드를 통해 추가할 배경지도 레이어 생성이 가능하며 load 함수의 마지막 부분에 추가합니다.

```
01  var lyr = new Xr.layers.TileMapLayer("basemap",
02      {
03          proxy: "http://222.237.78.208:8080/Xr",
04          url: "http://www.geoservice.co.kr/tilemap1",
05          ext: "png"
06      }
07  );
```

코드 8. 배경지도 레이어 생성

위의 레이어는 배경지도를 제공하는 공간서버로부터 타일맵 지도를 받아와 화면에 표시해 줄 수 있습니다.

다음으로 밀도 분석을 위한 입력 데이터로써 포인트 타입의 수치지도 레이어를 추가하겠습니다. 수치지도 레이어는 통계청에서 제공하는 전국의 인구수를 속성 데이터로 가지는 포인트 타입의 수치지도를 사용하

겠습니다. 다음 코드를 load 함수의 마지막에 추가합니다.

```
01  var shpLyr = new Xr.layers.ShapeMapLayer("population",
02        { url: "http://www.geoservice.co.kr:8080/Xr?layerName=population" });
03
04  shpLyr.needAttribute(true);
05  shpLyr.visibility().visibleByScale(true);
06  shpLyr.visibility().fromScale(0);
07  shpLyr.visibility().toScale(25000);
08
09  var label = shpLyr.label();
10  label.visibility().visibleByScale(true);
11  label.visibility().fromScale(0);
12  label.visibility().toScale(15000);
13  label.enable(false);
14
15  var labelTheme = label.theme();
16  labelTheme.symbol().strokeColor("#000000");
17  labelTheme.symbol().strokeWidth(2);
18
19  labelTheme.symbol().size(12);
20  labelTheme.symbol().fontFamily('맑은 고딕');
21  labelTheme.symbol().color("#ffff00");
22
23  var theme = shpLyr.theme();
24  var pen = theme.penSymbol();
25  var brush = theme.brushSymbol();
26
27  pen.color('#ff0000');
28  pen.width(2);
29  brush.color('#ffff00');
```

코드 9. 수치지도 레이어 생성

위의 코드를 살펴보면, 먼저 1번 코드는 인구수 집계구 데이터를 제공하는 수치지도를 사용하기 위해 ShapeMapLayer 객체를 생성하여 shpLyr이라는 변수에 담고 있습니다. 이 클래스의 생성자는 2개의 인자를 갖습니다. 첫 번째는 레이어에 대한 고유한 이름입니다. 이 이름 값을 이용하여 언제든지 해당 이름에 대한 레이어 객체를 참조할 수 있습니다. 두 번째 인자는 Object 타입의 객체로 이 객체에는 url이라는 프로퍼티를 가지고 있어야 합니다. url 프로퍼티를 통해 수치지도 레이어에 대한 연결 문자열(Connection String)을 지정합니다. 4번 코드는 수치지도 레이어가 항상 속성 데이터값을 가지도록 지정하고 있습니다. 5번 ~ 7번 코드는 수치지도 레이어가 보이는 축척 범위를 지정하고 있습니다. 9번 ~ 21번 코드는 수치지도 레이어에 대한 라벨을 설정하고 있는 코드입니다. 추후 가중치 값을 라벨로 표시할 것입니다. 23번 ~

29번 코드는 수치지도 레이어에 대한 포인트 데이터를 표시할 심벌을 지정하고 있습니다.

이제 앞서 생성한 배경지도 레이어 객체에 대한 lyr과 수치지도 레이어 객체에 대한 shpLyr을 레이어 관리자에 추가해야 합니다. 이에 대한 코드는 아래와 같습니다. load 함수의 마지막 줄에 추가합니다.

```
01  var lm = map.layers();
02
03  lm.add(lyr);
04  lm.add(shpLyr);
```

코드 10. 생성한 레이어 추가

이렇게 레이어를 최종적으로 추가 했으니 실행하면 지도가 표시될 것 같지만, 아직 한가지가 더 남아 있습니다. 그것은 레이어를 추가하고 난 뒤에 사용자에게 지도를 표시할 때 표시할 지도의 위치와 축척에 대한 것 입니다. 이에 대한 코드는 다음과 같으며 load 함수의 마지막 줄에 추가합니다.

```
01  map.onLayersAllReady(onLayersAllReady);
```

코드 11. 레이어 추가 완료시 호출되는 이벤트 지정

위의 코드에서 1번의 onLayersAllReady는 앞서 추가한 두 개의 레이어가 문제없이 추가가 되면 호출되는 이벤트 함수(Event Function 또는 Callback Function)로 onLayersAllReady를 지정하는 함수입니다. onLayersAllReady 함수는 다음과 같습니다.

```
01  function onLayersAllReady() {
02      var attributes = document.getElementById("attributes");
03      var option = document.createElement("option");
04      option.text = "사용 않함";
05      attributes.add(option);
06
07      var shpLyr = map.layers("population");
08      var fieldSet = shpLyr.attributeRowSet().fieldSet();
09      var cntFields = fieldSet.size();
10      for (var iField = 0; iField < cntFields; ++iField) {
11          var field = fieldSet.field(iField);
12          if (field.type() != Xr.data.FieldType.STRING) {
13              option = document.createElement("option");
14              option.text = field.name();
15              attributes.add(option);
16          }
17      }
18
19      var cm = map.coordMapper();
20      cm.moveTo(317782, 544590);
21      cm.zoomByMapScale(1534);
```

```
22
23        map.update();
24    }
```

코드 12. 레이어 추가 완료 시 호출되는 함수의 정의

위의 코드를 살펴보면, 먼저 2번 ~ 15번 코드는 밀도 분석을 위한 가중치 값으로 사용할 필드를 선택할 수 있는 컴보박스 UI에 필드명을 채우라는 것입니다. 가중치 값은 문자열이면 않되므로 12번 코드에서 문자열이 아닌 숫자형 데이터만을 갖는 필드에 대한 이름만을 추가하고 있습니다. 19번 코드는 지도의 이동, 확대, 축소, 회전 기능과 지도 좌표와 화면 좌표계 간의 변환 기능을 제공하는 CoordMapper 객체를 가져와 cm 이라는 변수에 저장하고 있습니다. 이 cm 변수를 이용하여 20번 ~ 21번에서 화면 상에 표시될 지도의 중심 좌표와 지도의 축척 지정하고 있습니다. 최종적으로 23번 코드에서 update 함수를 호출하여 지도를 그리도록 하고 있습니다. 실행해 보면 다음과 같습니다.

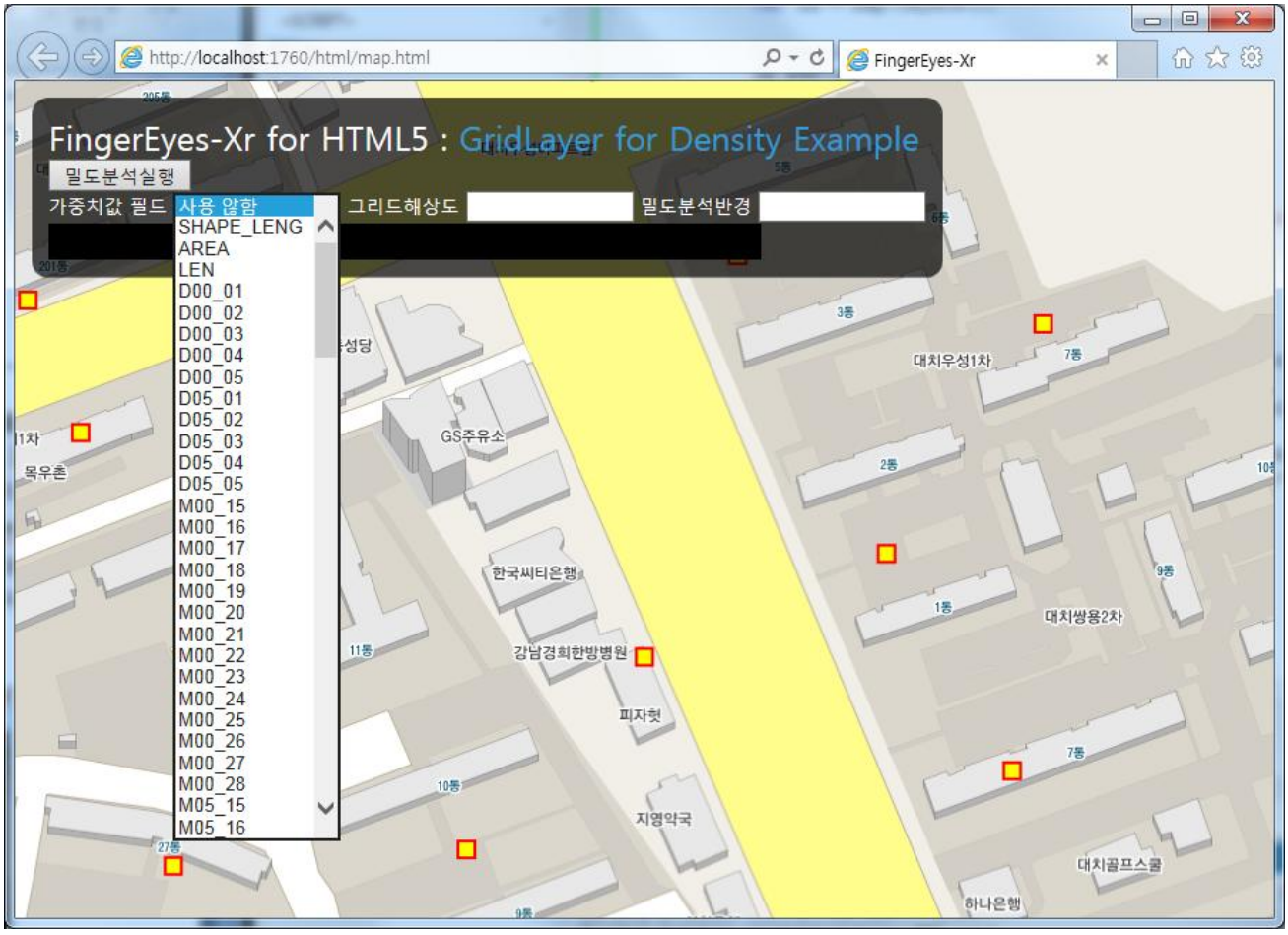

그림 5. 배경지도와 수치지도 레이어가 표시된 화면

실행해 보면 가중치값 필드에 필드명이 채워진 것을 볼 수 있고, 배경지도 레이어와 인구수 데이터가 담긴 포인트가 배경지도 위에 표시된 것을 볼 수 있습니다.

이제 지도를 조작할 수 있는 줌 레벨 컨트롤, 축척바 컨트롤을 화면에 배치해 보도록 하겠습니다. 먼저 축척바 컨트롤을 화면 상에 배치하도록 하겠습니다. 아래의 코드를 load 함수의 마지막 줄에 추가합니다.

```
01  var ctrl = new Xr.ui.ScaleBarControl("sbc", map);
02  map.userControls().add(ctrl);
```

코드 13. 축척바 컨트롤 추가

축척바 컨트롤 생성을 위해 ScaleBarControl 객체를 생성합니다. 첫번째 인자는 이 컨트롤에 대한 고유한 이름입니다. 이 이름을 통해 컨트롤에 접근이 가능합니다. 실행하면 아래의 그림처럼 화면 좌측 하단에 축척바 컨트롤이 표시됩니다.

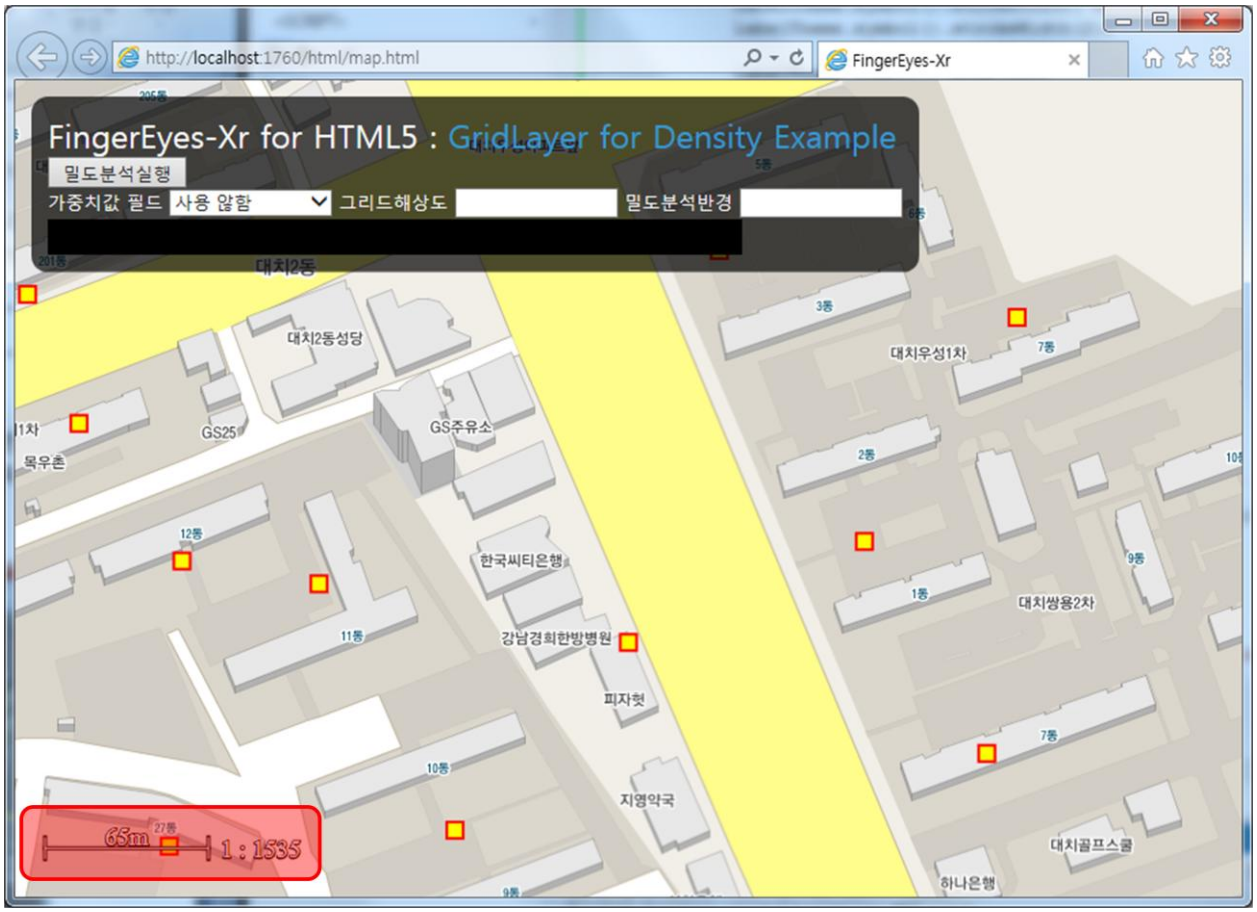

그림 6. 축척바 컨트롤 표시

다음으로 줌 레벨 컨트롤을 지도 화면에 표시하도록 하겠습니다. 이 줌 레벨 컨트롤을 표시하기 위해 load 함수의 마지막 줄에 아래의 코드를 추가합니다.

```
01  var ctrl2 = new Xr.ui.ZoomLevelControl("zlc", map);
02  ctrl2.mapScales([1533, 2682, 5746, 10726, 21988, 38307, 84274,
03      191531, 352416, 612897, 1379017, 2298362]);
04
05  map.userControls().add(ctrl2);
```

코드 14. 줌 레벨 컨트롤 추가

위의 코드에서 2번 코드에서 마우스 휠 조작이나 컨트롤 조작을 통해 반영되는 지도의 축척에 대한 분모 값을 배열 형태로 지정합니다. 실행하면 아래의 화면처럼 줌 레벨 컨트롤이 화면에 표시되는 것을 볼 수 있습니다.

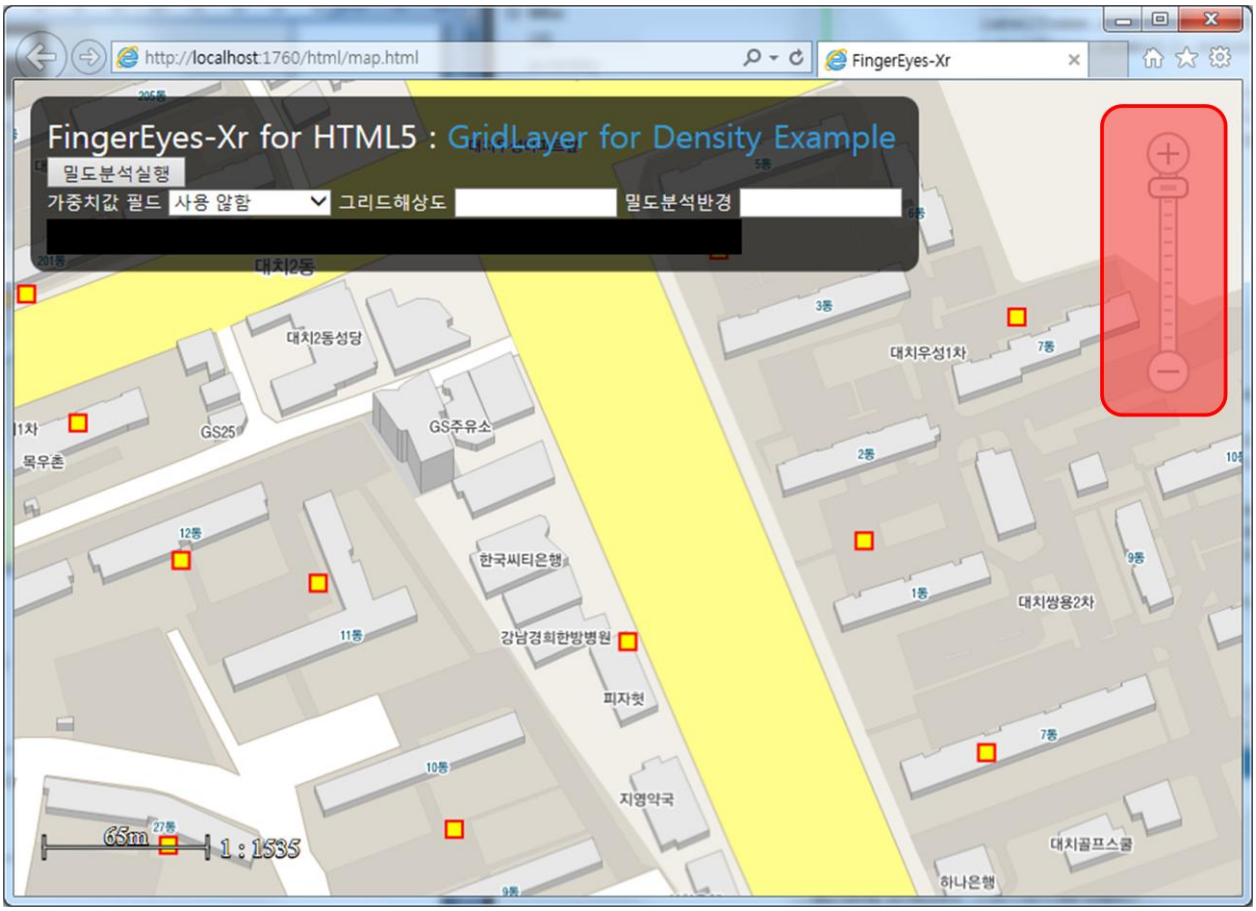

그림 7. 줌 레벨 컨트롤 추가

다음으로 밀도 분석에 대한 코드에 더 직접적인 내용에 대해 살펴보겠습니다.

실행 화면을 보면 그리드 해상도와 밀도분석 반경에 대한 입력 UI가 있습니다. 이 입력 UI는 사용자가 직접 입력할 수도 있지만 값을 자동으로 계산해 줄 수도 있습니다. 이 두 값을 계산하는 시점은 지도 뷰의 범위가 변경될 때가 가장 적당합니다. 지도 뷰가 변경될 때 발생하는 이벤트에서 이 두 값을 계산해 입력 UI에 값을 채워 보도록 하겠습니다.

다음 코드를 **코드 11. 레이어 추가 완료시 호출되는 이벤트 지정**의 마지막에 추가합니다.

```
01  map.addEventListener(Xr.Events.MapViewChanged, onMapViewChanged);
```

코드 15. 지도 뷰가 변경될 때 발생하는 이벤트 지정

MapViewChanged는 사용자가 지도의 축척을 변경하거나 지도를 이동할 경우마다 발생하는 이벤트로써 위의 코드는 이러한 이벤트에 대한 호출 함수로 onMapViewChanged를 호출하라고 지시하는 것입니다. onMapViewChanged 함수는 다음과 같습니다.

```
01  function onMapViewChanged(e) {
02      var coordMapper = map.coordMapper();
03      var mbr = coordMapper.viewportMBR();
```

```
04      var resolution = coordMapper.metersPerOnePixel() * 1;
05      var radius = coordMapper.metersPerOnePixel() * 200;
06
07      var txtResolution = document.getElementById("resolution");
08      var txtRadius = document.getElementById("radius");
09
10      txtResolution.value = resolution.toFixed(4);
11      txtRadius.value = radius.toFixed(4);
12  }
```

코드 16. 지도 뷰가 변경될 때 발생하는 이벤트 함수

위의 코드를 보면 그리드해상도와 밀도분석반경을 계산하여 각 입력 UI에 값을 지정하고 있습니다. 그리드 해성도를 계산하기 위한 코드는 4번입니다. 1 Pixel에 대한 지도 단위값을 사용하고 있고, 밀도분석반경은 그리드 해상도 값에 200을 곱한 값을 사용하고 있습니다. 이제 실행하고 지도를 축소나 확대해 보면 그리드 해상도와 밀드분석반경에 해당하는 입력 UI에 값이 채워지는 것을 볼 수 있습니다.

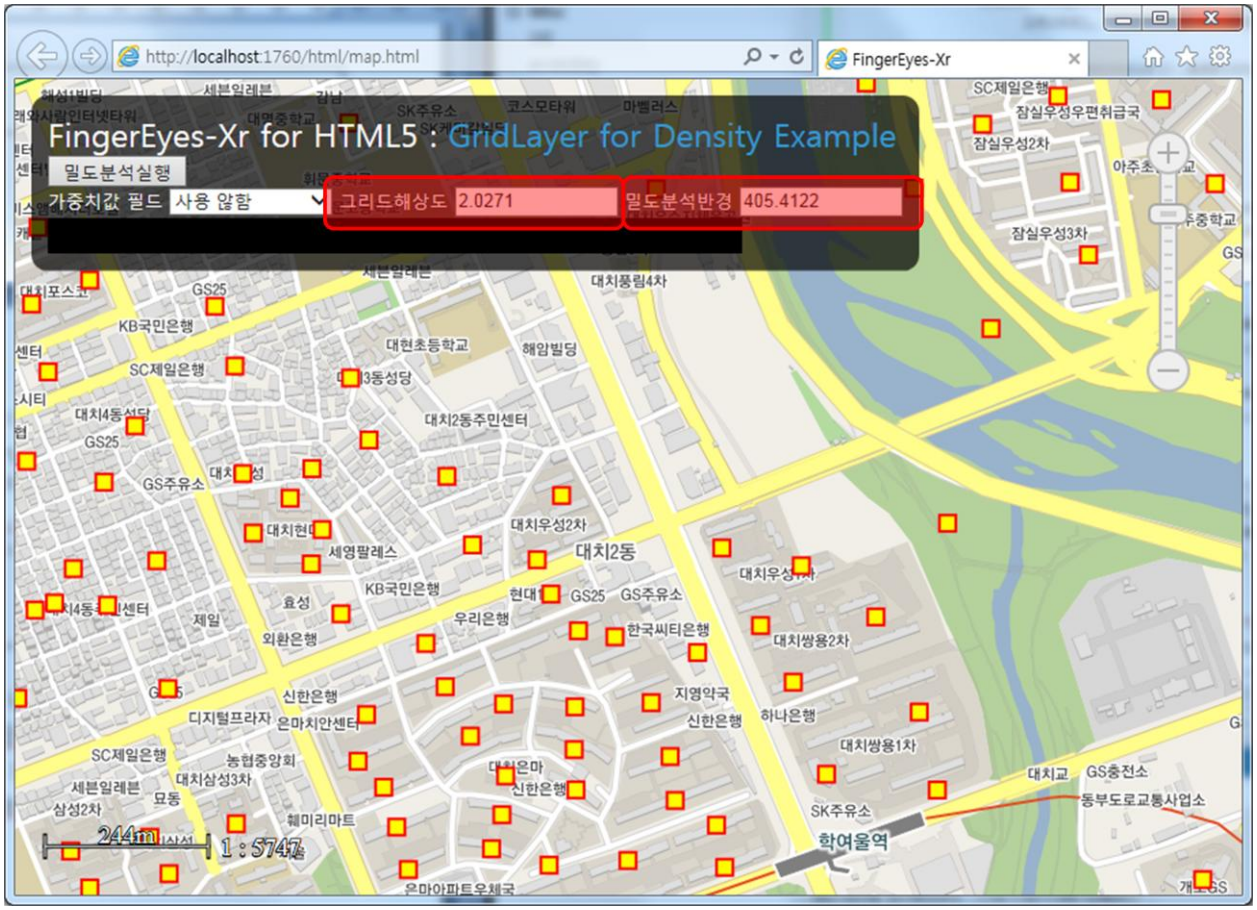

그림 8. 밀도 분석에 필요한 인자값에 대한 UI 반영

앞서 가중치값 필드를 나타내는 컴보박스 UI에 집계구에 대한 인구수를 나타내는 수치지도의 필드명을 채웠습니다. 이 컴보박스에 대해 사용자가 선택 항목을 변경할 때 발생하는 이벤트를 지정하고 이 이벤트 함수에 대해 살펴보겠습니다. 이미 이 컴보박스의 onchange 이벤트에는 onWeightedValueFieldChange 함수가 지정되어 있습니다. onchange 이벤트는 사용자가 컴보박스의 항목을 변경할 때 발생하는 이벤트이며 이 이벤트에 대한 호출 함수인 onWeightedValueFieldChange 함수는 다음과 같습니다.

```
01  function onWeightedValueFieldChange() {
02      var attributes = document.getElementById("attributes");
03      var selectedIdx = attributes.selectedIndex;
04      var shpLyr = map.layers("population");
05      var label = shpLyr.label();
06
07      if (selectedIdx == 0) {
08          label.enable(false);
09      } else {
10          label.enable(true);
11          var fieldName = attributes.options[selectedIdx].value;
12          label.formatter().fieldName(fieldName)
13      }
14
15      map.update();
16  }
```

코드 17. 가중치값 필드에 대한 컴보박스 UI의 onchange 이벤트

위의 함수를 살펴보면, 2번 ~ 3번 코드를 통해 컴보박스의 선택 항목에 대한 인덱스를 얻어 selectedIdx 변수에 저장합니다. 그리고 4번 ~ 5번 코드를 통해 밀도 분석 대상이 되는 인구수 수치지도 레이어의 라벨 객체를 얻어 label 변수에 저장합니다. 7번 ~ 13번 코드는 만약 selectedIndx 값이 0이면 가중치 값을 사용하지 않는다는 의미이므로 라벨 표시를 비활성화하고 1이 아닌 값이라면 라벨 표시를 활성화하면서 라벨 표시에 사용되는 필드명을 사용자가 선택한 필드명으로 지정합니다. 15번 코드는 지도를 다시 그리라는 것입니다. 실행하여 컴보 박스를 변경해 보면 아래의 그림처럼 선택된 필드명에 저장된 값이 지도 위에 라벨로 표시되는 것을 볼 수 있습니다.

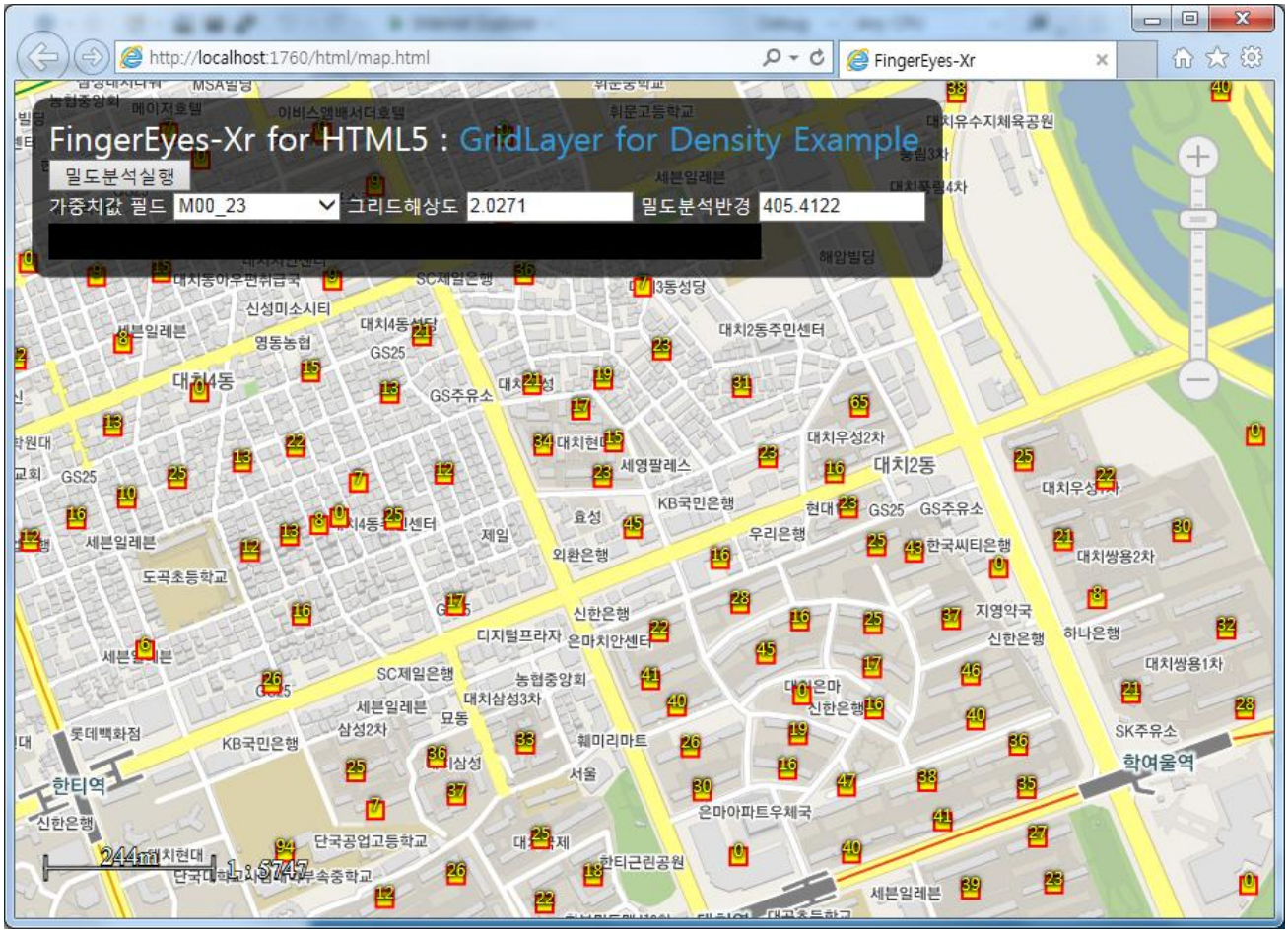

그림 9. 가중치값 필드에 대한 라벨 표시

이제 밀도분석을 위해 필요한 인자값들의 준비가 끝났습니다. 밀도분석실행 버튼을 클릭하면 준비된 인자값을 이용하여 그리드 레이어를 생성하고 밀도 분석을 수행하게 됩니다. 밀도분석실행 버튼의 onclick 이벤트에는 onDensity 함수가 지정되어 있습니다. 이 함수는 다음과 같습니다.

```
01  function onDensity() {
02      var coordMapper = map.coordMapper();
03      var mbr = coordMapper.viewportMBR();
04
05      runDensity(mbr);
06  }
```

코드 18. 밀도분석 실행 버튼 클릭 이벤트 함수

위의 코드를 보면 3번에서 현재 지도 화면에 해당하는 MBR을 얻어와 mbr 변수에 저장하고 있습니다. 그리고 5번 코드에서 runDensity 함수에 mbr 인자를 입력하고 호출하고 있습니다. 이는 밀도분석의 범위를 현재 지도 화면에 표시된 인구수 포인트로 제한하는 것입니다. runDensity 함수는 다음과 같습니다.

```
01  function runDensity(mbr) {
02      var coordMapper = map.coordMapper();
03      var txtResolution = document.getElementById("resolution");
04      var txtRadius = document.getElementById("radius");
```

```
05      var resolution = parseFloat(txtResolution.value);
06      var radius = parseFloat(txtRadius.value);
07
08      var gridLyr = new Xr.layers.GridLayer("grid",
09          {
10              mbr: new Xr.MBR(mbr.minX - radius, mbr.minY - radius,
11                      mbr.maxX + radius, mbr.maxY + radius),
12              resolution: resolution
13          }
14      );
15
16      map.layers().remove("grid");
17      map.layers().add(gridLyr);
18
19      var shapeLyr = map.layers("population");
20      var label = shapeLyr.label();
21      var clrTbl = new Xr.ColorTable(9);
22
23      clrTbl.set(0, 0, 255, 0, 40);
24      clrTbl.set(5, 255, 255, 0, 190);
25      clrTbl.set(8, 255, 0, 0, 255);
26
27      if (clrTbl.build()) {
28          var fieldName = label.enable() ?
29              label.formatter().fieldName() : undefined;
30          if (!gridLyr.densityByLayer(shapeLyr, radius, clrTbl.colors(),
31              fieldName, progressCallback, "js/density.js")) {
32              alert("Error setting CellValues By ShapeMapLayer");
33          } else {
34              var btn = document.getElementById("density");
35              btn.disabled = true;
36          }
37      } else {
38          alert("색상 테이블 생성 실패");
39      }
40  }
```

코드 19. 밀도 분석 실행 함수

위의 함수를 살펴보면, 5번 코드에서는 resolution 변수에 그리드 해상도 값을 저장하고 6번 코드에서는 radius 변수에 밀도분석반경 값을 저장합니다. 그리고 8번 코드에서 그리드 레이어를 생성하기 위해 GridLayer 클래스의 생성자를 호출하고 있습니다. 이 GridLayer의 생성자는 2개의 인자를 갖습니다. 첫번째는 레이어의 고유한 이름이고 두번째는 그리드 레이어의 내부 데이터, 정확히는 셀(Cell) 데이터를 구성

하기 위한 속성값을 담은 객체입니다. 이 객체에는 그리드 레이어가 참조하는 공간상의 범위인 mbr 속성과 그리드 해상도에 대한 resolution 속성입니다. mbr 속성에는 runDensity 함수에 전달한 mbr 인자에 대해 밀도분석반경 만큼 확장한 범위를 계산하여 지정합니다. 16번 코드에서는 이전에 동일한 이름의 그리드 레이어가 추가되어져 있다면 제거하고 17번 코드에서 다시 추가합니다. 19번 코드는 밀도 분석 대상이 되는 수치지도 레이어를 가져오고 20번 코드는 이 레이어의 라벨 객체를 얻어옵니다. 21번 코드는 밀도 분석이 완료되면 그 결과를 이미지화할 때 필요한 색상 테이블을 생성하는데, 총 9개의 색상 값을 담을 수 있는 ColorTable 객체를 생성합니다. 23번 ~ 25번 코드는 ColorTable 객체가 관리하고 있는 9개의 색상 중 첫번째(0값이 첫번째임)와 여섯번째 그리고 마지막인 아홉번째의 색상값을 각각 RGBA로써 (0,255,0, 40)과 (255, 255, 0, 190) 그리고 (255, 0, 0, 255)로 지정합니다. 27번 코드는 이렇게 지정한 3개의 색상을 통해 아직 지정하지 않은 중간 단계의 색상을 자동으로 생성하기 위해 ColorTable 객체의 build 함수를 호출하고, 성공하면 28번에서 밀도 분석을 위해 사용할 가중치 필드명을 얻어옵니다. 가중치로 사용할 필드는 수치지도 레이어의 라벨 필드명과 동일하다는 점에 착안하였습니다. 그리고 30번에서 실제 밀도분석을 수행하기 위해 그리드 레이어의 densityByLayer 함수를 호출합니다. 이 함수는 6개의 인자를 갖습니다. 첫번째는 집계구에 대한 인구수를 가지고 있는 수치지도 레이어 객체이고 두번째는 밀도분석반경 값, 세번째는 밀도 분석이 완료되었을 때 그 결과를 이미지화 하기 위한 색상값들이 저장된 배열 객체, 네번째는 가중치값을 위한 필드명으로 이 필드명을 undefined으로 주면 가중치를 사용하지 않습니다. 그리고 다섯번째는 밀도분석이 진행될 때 진행율에 대한 피드백을 주기 위한 콜백(Callback) 함수입니다. 그리고 마지막으로 여섯번째는 밀도분석에 사용되는 Plug-In 소스코드 경로입니다. 여기서 여섯번째 인자인 밀도분석에 사용되는 Plug-In 소스 코드로 지정한 density.js 파일의 내용은 아래와 같습니다. 이 density.js 파일의 위치는 map.html이 있는 폴더에 js 폴더를 만들고 이 js 폴더 안에 존재해야 합니다.

```
01  /**
02   * @desc FingerEyes-Xr for HTML5을 위한 가중치값을 고려한 밀도 분석 플러그인(Plug-In)
03   * @version 1.0
04   * @copyright [(주)지오서비스]{@link http://www.geoservice.co.kr}
05   * @license LGPL
06   */
07
08  self.onmessage = function (event) {
09      var postData = event.data;
10      var cntRows = postData.countRows;
11      var resolution = postData.resolution;
12      var radius = postData.radius;
13      var mbr = postData.mbr;
14      var mbrMinX = mbr.minX;
15      var mbrMinY = mbr.minY;
16      var cntColumns = postData.countColumns;
17      var cells = postData.cells;
18      var minValue = Number.MAX_VALUE;
19      var maxValue = -Number.MAX_VALUE;
20      var data = postData.data;
```

```
21      var colorTable = postData.colorTable;
22      var imgRawData = postData.imgRawData;
23
24      // analysis density
25      var cntData = data.length;
26      if (data.length > 0) {
27          for (var i = 0; i < cntData; i++) {
28              var progress = (i / cntData) * 70; // 70%
29              self.postMessage(progress);
30
31              var datum = data[i];
32              var cx = datum.x;
33              var cy = datum.y;
34              var value = datum.value;
35              var startX = cx - radius;
36              var endX = cx + radius;
37              var startY = cy - radius;
38              var endY = cy + radius;
39              var s = radius / 4; // 나누는 값이 클수록 경계가 더 매끄럽게 표현됨
40
41              for (var x = startX; x <= endX; x += resolution) {
42                  for (var y = startY; y <= endY; y += resolution) {
43                      var r = Math.sqrt(Math.pow(x - cx, 2.0) + Math.pow(y - cy, 2.0));
44                      if (r > radius) continue;
45
46                      var fx = (1 / (Math.sqrt(2 * Math.PI) * s))
47                          * Math.pow(Math.E, -0.5 * Math.pow(r / s, 2.0));
48                      var v = value * fx;
49
50                      var row = cntRows - parseInt(Math.floor((y - mbrMinY) / resolution)) - 1;
51                      var column = parseInt(Math.floor((x-mbrMinX) / resolution));
52                      var cellIdx = row * cntColumns + column;
53                      var prevValue = cells[cellIdx];
54                      if (prevValue != undefined) {
55                          if (isNaN(prevValue)) prevValue = 0;
56                          var lastValue = prevValue + v;
57                          cells[cellIdx] = lastValue;
58
59                          if (lastValue < minValue) minValue = lastValue;
60                          if (lastValue > maxValue) maxValue = lastValue;
61                      }
62                  }
```

```
63              }
64          }
65          // .
66
67          // create bitmap
68          var countIntv = colorTable.length;
69          var lenIntv = (maxValue - minValue) / countIntv;
70          var totalCells = cntRows * cntColumns;
71
72          for (var row = 0; row < cntRows; row++) {
73              for (var col = 0; col < cntColumns; col++) {
74                  var progress = ((row * cntColumns + col) / totalCells) * 30; // 30%
75                  if (progress - parseInt(progress) == 0) {
76                      self.postMessage(70 + progress);
77                  }
78
79                  var cellValue = cells[row * cntColumns + col];
80                  if (cellValue && !isNaN(cellValue)) {
81                      var xx = cellValue - minValue;
82                      var indexIntv = Math.floor((cellValue - minValue) / lenIntv);
83
84                      if (indexIntv == countIntv) indexIntv--;
85                      var clr = colorTable[indexIntv];
86
87                      imgRawData[((cntColumns * row) + col) * 4] = clr.r;
88                      imgRawData[((cntColumns * row) + col) * 4 + 1] = clr.g;
89                      imgRawData[((cntColumns * row) + col) * 4 + 2] = clr.b;
90                      imgRawData[((cntColumns * row) + col) * 4 + 3] = clr.a;
91                  }
92              }
93          }
94
95          self.postMessage(100);
96          // .
97
98          self.postMessage(
99              {
100                 imgRawData: imgRawData,
101                 cells: cells,
102                 minValue: minValue,
103                 maxValue: maxValue
104             }
```

```
105             );
106         }
107 }
```

코드 20. 밀도 분석 plug-in 소스 코드

densityByLayer 함수의 다섯번째 인자인 밀도분석의 진행율에 대한 피드백을 주는 콜백(Callback) 함수인 progressCallback 함수는 다음과 같습니다.

```
01 function progressCallback(/* number */ percentage) {
02     progressbar.innerHTML = parseInt(percentage) + '%';
03     progressbar.style.width = (percentage * 5) + 'px';
04 
05     if (percentage == 100) {
06         var btn = document.getElementById("density");
07         btn.disabled = false;
08 
09         var yes = confirm("밀도 분석이 완료되었습니다. 분석된 영역으로 이동하시겠습니까?")
10         if (yes) {
11             var gridLyr = map.layers("grid");
12             var mbr = gridLyr.MBR();
13             map.coordMapper().zoomByMBR(mbr);
14         }
15 
16         map.update();
17     }
18 }
```

코드 21. 밀도 분석 진행율에 대한 피드백을 받는 콜백함수

progressCallback 함수는 하나의 인자를 받습니다. 바로 0 ~ 100% 범위의 진행율입니다. id 값이 progressbar인 DIV를 이용해 마치 ProgressBar와 같은 UI를 흉내 내고 있습니다. 5번 코드에서 진행율이 100%인지 검사하고 9번 코드에서 사용자에게 밀도 분석이 완료되었다는 메시지와 함께 분석 영역으로 이동할지를 묻습니다. 밀도 분석은 시간이 많이 소요될 수 있습니다. (참고로 IE 버전 11과 Chrome 버전 40의 밀도 분석 속도는 10배 이상 차이가 남). FingerEyes-Xr for HTML5는 밀도 분석 진행 중에도 지도의 이동, 확대, 축소와 같은 작업을 진행할 수 있습니다. 만약 사용자가 밀도 분석 중에 다른 위치로 지도를 이동했을 때 밀도 분석이 완료되면서 해당 분석 위치로 이동할지를 묻는 편의성을 제공하는 것이 효과적일 것입니다. 코드 11번 ~ 13번이 바로 해당 분석 위치로 이동하는 코드입니다. 16번은 지도를 다시 그리기 위한 함수 호출입니다. 실행하고 밀도분석실행 버튼을 클릭하면 다음과 같은 결과가 표시되는 것을 확인할 수 있습니다.

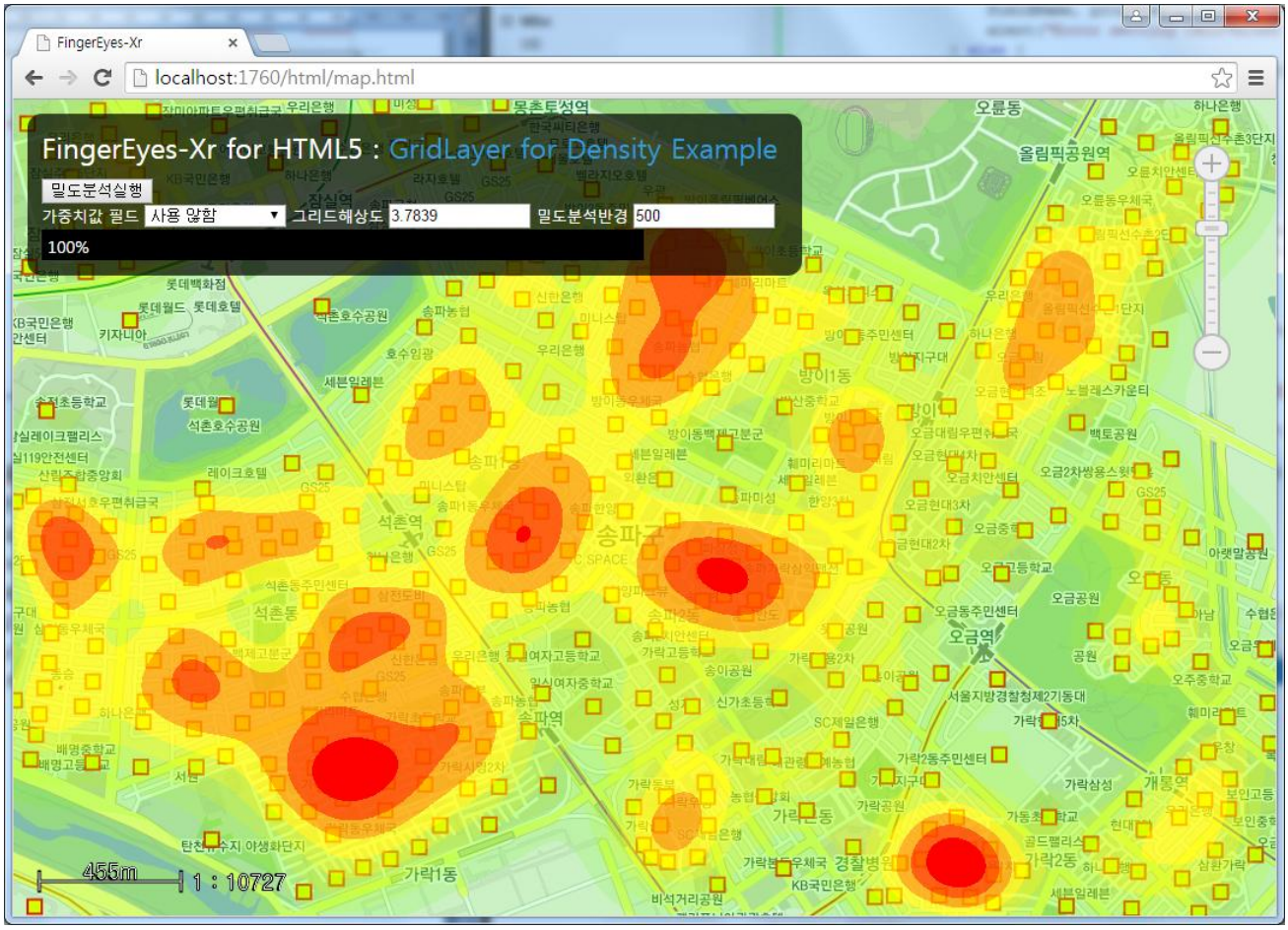

그림 10. 밀도 분석 결과 이미지

이상으로 그리드 레이어를 이용한 밀도 분석을 FingerEyes-Xr for HTML5에서 구현해 보는 내용에 대해 살펴보았습니다. 아래는 지금까지 작성한 전체 소스 코드입니다.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <style>
    body
    {
        margin:0px;
        padding:0px;
    }

    #mainLayout
    {
        width:100%;
        height:100%;
        border:none;
    }

    #mapDiv
    {
        top:0px;
        left:0px;
        position:relative;
        width:100%;
        height:100%;
        border:none;
```

```
        overflow:hidden;
    }

    #title
    {
        top:12px;
        left:12px;
        padding: 12px;
        position:absolute;
        background:rgba(0,0,0,0.7);
        border:none;
        overflow:auto;
        border-radius: 12px;
        font-size: 24px;
        color: #ffffff;
        font-family: "맑은 고딕";
    }

    #progressbar-bg
    {
        width: 510px;
        background: gray;
    }

    #progressbar
    {
        height: 16px;
        margin: 1px 0px;
        color: #fff;
        padding: 5px;
        background: rgb(0,0,0);
        font-size: 14px;
    }
    </style>

    <title>FingerEyes-Xr</title>

    <script src="http://www.geoservice.co.kr/z/Xr.min.1.0.js"></script>
    <script type="text/javascript">
        var map = null;

        function load() {
            map = new Xr.Map("mapDiv", {});

            var lyr = new Xr.layers.TileMapLayer("basemap",
                {
                    proxy: "http://222.237.78.208:8080/Xr",
                    url: "http://www.geoservice.co.kr/tilemap1",
                    ext: "png"
                }
            );

            var shpLyr = new Xr.layers.ShapeMapLayer("population",
                { url:
"http://www.geoservice.co.kr:8080/Xr?layerName=population" });

            shpLyr.needAttribute(true);
            shpLyr.visibility().visibleByScale(true);
            shpLyr.visibility().fromScale(0);
            shpLyr.visibility().toScale(25000);

            var label = shpLyr.label();
            label.visibility().visibleByScale(true);
            label.visibility().fromScale(0);
```

```
        label.visibility().toScale(15000);
        label.enable(false);

        var labelTheme = label.theme();
        labelTheme.symbol().strokeColor("#000000");
        labelTheme.symbol().strokeWidth(2);

        labelTheme.symbol().size(12);
        labelTheme.symbol().fontFamily('맑은 고딕');
        labelTheme.symbol().color("#ffff00");

        var theme = shpLyr.theme();
        var pen = theme.penSymbol();
        var brush = theme.brushSymbol();

        pen.color('#ff0000');
        pen.width(2);
        brush.color('#ffff00');

        var lm = map.layers();

        lm.add(lyr);
        lm.add(shpLyr);

        map.onLayersAllReady(onLayersAllReady);
        map.addEventListener(Xr.Events.MapViewChanged, onMapViewChanged);

        var ctrl = new Xr.ui.ScaleBarControl("sbc", map);
        map.userControls().add(ctrl);

        var ctrl2 = new Xr.ui.ZoomLevelControl("zlc", map);
        ctrl2.mapScales([1533, 2682, 5746, 10726, 21988, 38307, 84274,
            191531, 352416, 612897, 1379017, 2298362]);

        map.userControls().add(ctrl2);


    }

    function onMapViewChanged(e) {
        var coordMapper = map.coordMapper();
        var mbr = coordMapper.viewportMBR();
        var resolution = coordMapper.metersPerOnePixel() * 1;
        var radius = coordMapper.metersPerOnePixel() * 200;

        var txtResolution = document.getElementById("resolution");
        var txtRadius = document.getElementById("radius");

        txtResolution.value = resolution.toFixed(4);
        txtRadius.value = radius.toFixed(4);
    }

    function onLayersAllReady() {
        var attributes = document.getElementById("attributes");
        var option = document.createElement("option");
        option.text = "사용 않함";
        attributes.add(option);

        var shpLyr = map.layers("population");
        var fieldSet = shpLyr.attributeRowSet().fieldSet();
        var cntFields = fieldSet.size();
        for (var iField = 0; iField < cntFields; ++iField) {
            var field = fieldSet.field(iField);
```

```
            if (field.type() != Xr.data.FieldType.STRING) {
                option = document.createElement("option");
                option.text = field.name();
                attributes.add(option);
            }
        }

        var cm = map.coordMapper();
        cm.moveTo(317782, 544590);
        cm.zoomByMapScale(1534);

        map.update();
    }

    function onWeightedValueFieldChange() {
        var attributes = document.getElementById("attributes");
        var selectedIdx = attributes.selectedIndex;
        var shpLyr = map.layers("population");
        var label = shpLyr.label();

        if (selectedIdx == 0) {
            label.enable(false);
        } else {
            label.enable(true);
            var fieldName = attributes.options[selectedIdx].value;
            label.formatter().fieldName(fieldName)
        }

        map.update();
    }

    function onDensity() {
        var coordMapper = map.coordMapper();
        var mbr = coordMapper.viewportMBR();

        runDensity(mbr);
    }

    function runDensity(mbr) {
        var coordMapper = map.coordMapper();
        var txtResolution = document.getElementById("resolution");
        var txtRadius = document.getElementById("radius");
        var resolution = parseFloat(txtResolution.value);
        var radius = parseFloat(txtRadius.value);

        var gridLyr = new Xr.layers.GridLayer("grid",
            {
                mbr: new Xr.MBR(mbr.minX - radius, mbr.minY - radius,
                        mbr.maxX + radius, mbr.maxY + radius),
                resolution: resolution
            }
        );

        map.layers().remove("grid");
        map.layers().add(gridLyr);

        var shapeLyr = map.layers("population");
        var label = shapeLyr.label();
        var clrTbl = new Xr.ColorTable(9);

        clrTbl.set(0, 0, 255, 0, 40);
        clrTbl.set(5, 255, 255, 0, 190);
        clrTbl.set(8, 255, 0, 0, 255);
```

```
            if (clrTbl.build()) {
                var fieldName = label.enable() ?
                    label.formatter().fieldName() : undefined;
                if (!gridLyr.densityByLayer(shapeLyr, radius, clrTbl.colors(),
                    fieldName, progressCallback, "js/density.js")) {
                    alert("Error setting CellValues By ShapeMapLayer");
                } else {
                    var btn = document.getElementById("density");
                    btn.disabled = true;
                }
            } else {
                alert("색상 테이블 생성 실패");
            }
        }

        function progressCallback(/* number */ percentage) {
            progressbar.innerHTML = parseInt(percentage) + '%';
            progressbar.style.width = (percentage * 5) + 'px';

            if (percentage == 100) {
                var btn = document.getElementById("density");
                btn.disabled = false;

                var yes = confirm("밀도 분석이 완료되었습니다. 분석된 영역으로
이동하시겠습니까?")
                if (yes) {
                    var gridLyr = map.layers("grid");
                    var mbr = gridLyr.MBR();
                    map.coordMapper().zoomByMBR(mbr);
                }

                map.update();
            }
        }


    </script>

</head>
<body onload="load()">
    <div id="mainLayout">
        <div id="mapDiv"></div>
        <div id="title">
            FingerEyes-Xr for HTML5 : <font color="#349bd6">GridLayer for Density
Example</font>
            <br />
            <input type="button" id="density" value="밀도분석실행"
onclick="onDensity();">
            <br />
            <font size="2" color="#ffffff"><span>가중치값 필드</span>
            <SELECT id='attributes' size='1' style="width:120px"
onchange="onWeightedValueFieldChange()"></SELECT>
            <span>그리드해상도</span> <input type="text" style="width:120px"
id="resolution">
            </span>밀도분석반경 </span><input type="text" style="width:120px"
id="radius">
            </font>
            <div id="progressbar-bg"><div id="progressbar"></div></div>
        </div>
    </div>
</body>
```

```
</html>
```