FingerEyes-Xr for HTML5 Tutorials - 08

수치지도에 대한 사용자 정의 라벨

2015년 3월 2일, 1차 배포

㈜지오서비스

수치지도에 대한 사용자 정의 라벨(Label)

여기서 말하는 수치 지도란 서버로부터 수치값 형태의 좌표값과 속성값을 받아 FingerEyes-Xr에서 그릴 수 있는 데이터를 가지고 있는 지도를 의미합니다. FingerEyes-Xr에서 활용할 수 있는 수치지도는 OGC 표준 중에 하나인 WFS에서 지도를 구성할 수 있는 WFSLayer 클래스와 Custom Binary 포맷으로부터 지도를 구성할 수 있는 ShapeMapLayer 클래스를 통해 사용이 가능합니다.

이 글은 WFSLayer와 ShapeMapLayer에서 속성값에 따라 라벨의 그리기 심벌은 물론 라벨 텍스트를 제어하는 방법에 대해 설명합니다. 이 기능을 이용하면 속성값에 따라 라벨의 폰트 크기를 조절할 수 있으며, 여러 개의 필드값을 조합하거나 필드값에서 필요한 값만을 추출하여 라벨의 텍스트로 표현할 수 있습니다.

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.

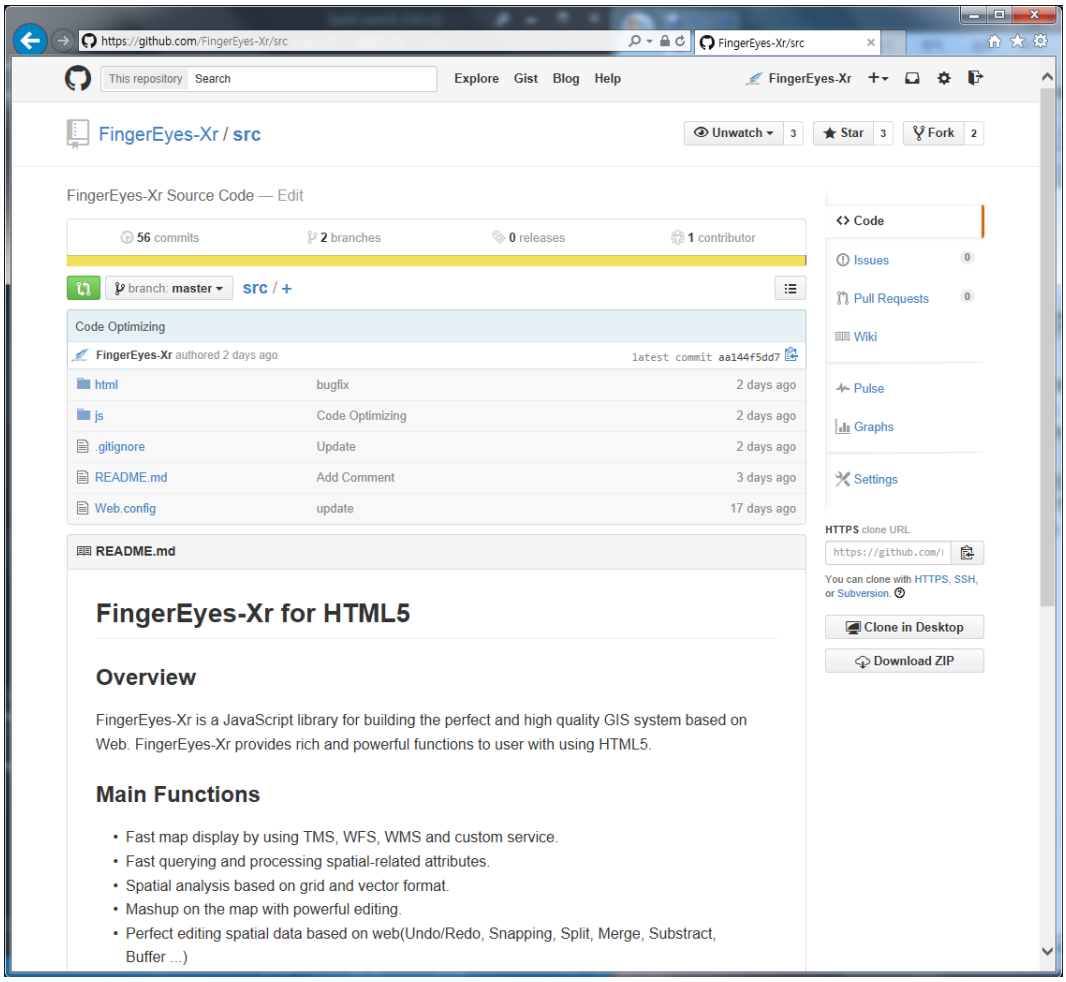


그림 1. FingerEyes-Xr for HTML에 대한 GitHub

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 WFSLayer와 ShapeMapLayer와 같은 수치지도 레이어의 라벨에 대한 그리기 심벌을 속성값에 따라 다양하게 적용하고 필드값에서 필요한 값만을 추출하여 라벨의 텍스트로 지정하는 코드를 작성해 보겠습니다. 여기서는 ShapeMapLayer를 사용했으며 지적필지의 종류에 따라 라벨의 그리기 색상을 지정하는 것에 대한 주제로 설명을 하겠습니다. 즉, 아래처럼 지적필지의 종류가 답, 공, 잡, 전 등에 따라 라벨의 텍스트 색상을 결정하겠습니다. 아래의 그림에서 EEB500과 같은 16진수 표기는 RGB 색상값을 의미합니다. 참고로 WFSLayer도 동일한 API를 사용하여 속성값으로 라벨의 그리기 심벌을 설정할 수 있습니다.

그림 2. 지적 필지 종류에 따른 라벨의 텍스트 색상값

이제 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 웹 브라우저에서 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 입력 될 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 :
06              <font color="#349bd6">Custom Label Example</font>
07          </div>
08      </div>
09  </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.

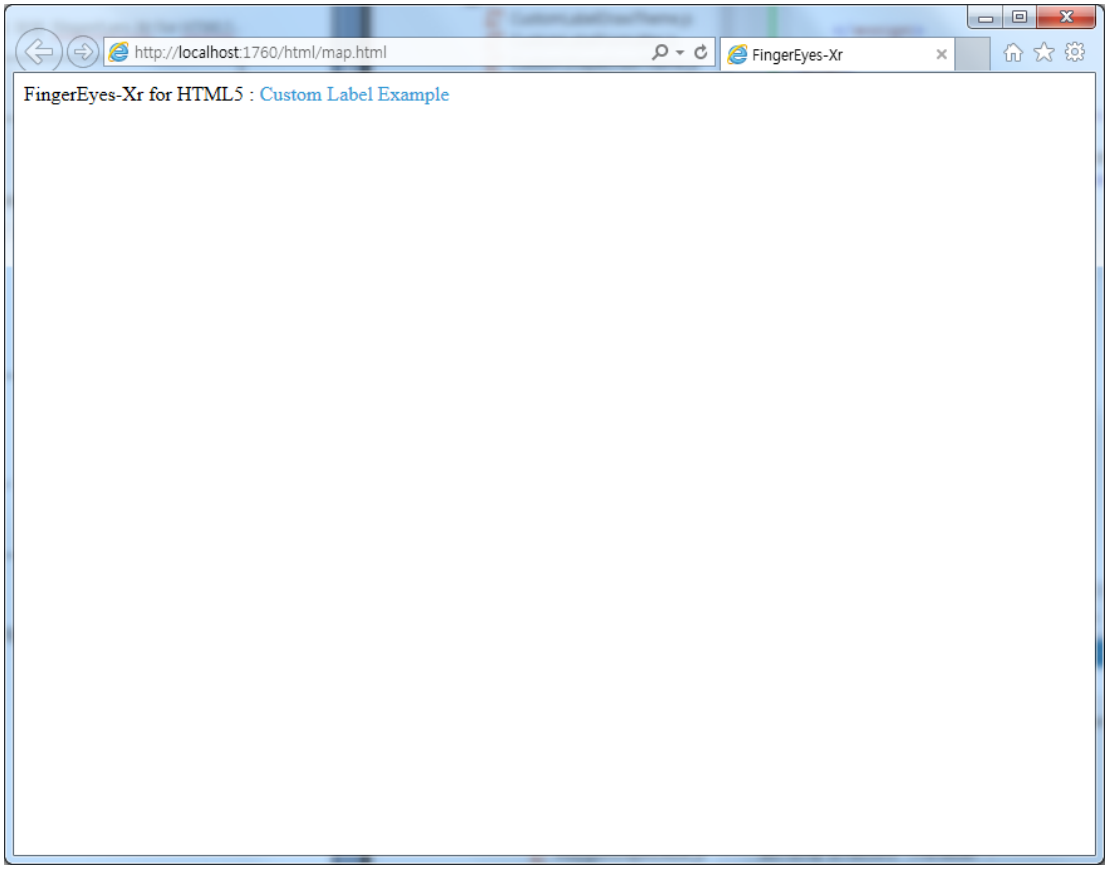


그림 3. 기본 UI 구성에 대한 실행 화면

이제 만든 페이지에 스타일을 적용해 보도록 하겠습니다. <head> 바로 밑에 아래의 스타일 코드를 추가합니다.

```
<style>
    body {
        margin:0px;
        padding:0px;
    }

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

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

    #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: "Arial";
    }
</style>
```

코드 4. UI에 대한 스타일 적용

스타일이 적용된 페이지는 다음과 같습니다.

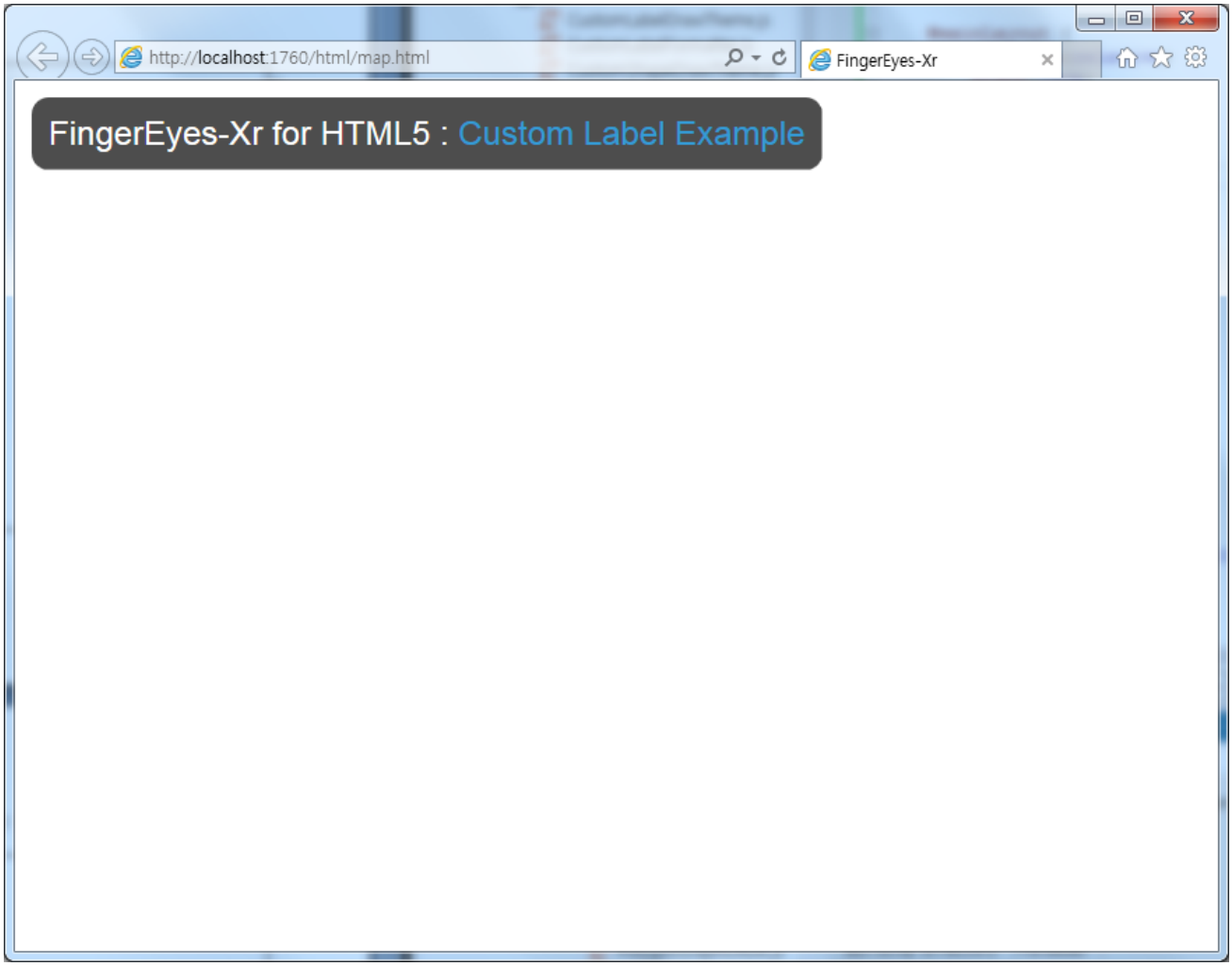


그림 4. 스타일이 적용된 UI

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.

이제 수치지도를 표시하기에 앞서 항공영상 지도를 배경지도로 깔고 그 위에 수치지도를 표시하는 것으로 하겠습니다. 항공영상 지도를 표시는 다음 코드를 통해 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://222.237.78.208:8080/muan_tilemaps",
05          ext: "jpg"
06      }
07  );
```

코드 8. 항공영상 배경지도 레이어 생성

위의 레이어는 항상영상 지도를 제공하는 공간서버로부터 항공영상 지도를 받아와 화면에 표시해 주는 코드입니다.

이제 지적필지를 나타내는 수치지도 레이어를 표시하겠습니다. 다음 코드를 load 함수의 마지막에 추가합니다.

```
01  var jibunLyr = new Xr.layers.ShapeMapLayer("jibun",
02      {
03          url: "http://222.237.78.208:8080/Xr?layerName=BML_PARC_AS"
04      }
05  );
```

코드 9. 수치지도 레이어 생성

위의 코드를 살펴보면, 먼저 1번 코드는 Custom Binary 포맷을 이용하는 수치지도를 사용하기 위해 ShapeMapLayer 객체를 생성하여 jibunLyr이라는 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.

이제 앞서 생성한 항공영상 레이어 객체에 대한 lyr과 수치지도 레이어 객체에 대한 jibunLyr을 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(jibunLyr);
```

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(function () {
02      var cm = map.coordMapper();
03
04      cm.moveTo(151531, 246679);
05      cm.zoomByMapScale(766);
06
07      map.update();
08  });
```

코드 11. 레이어 추가 완료가 되면 지도의 표출

위의 코드에서 1번의 onLayersAllReady는 앞서 추가한 두 개의 레이어가 문제없이 추가가 되면 호출되는 이벤트 함수(Event Function 또는 Callback Function)를 지정하는 함수입니다. 편의 상 익명 함수를 사용하여 이벤트를 지정 하였습니다. 해당 익명 함수의 코드 부분을 살펴보면, 먼저 2번 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 4번 코드에서 화면 상에 표시될 지도의 중심 좌표를 지정하고 5번 코드에서 지도의 축척을 1:766으로 지정하기 위해 축척의 분모값인 766을 인자로 호출하고 있습니다. 최종적으로 7번 코드에서 update 함수를 호출하여 지도를 그리도록 하고 있습니다.

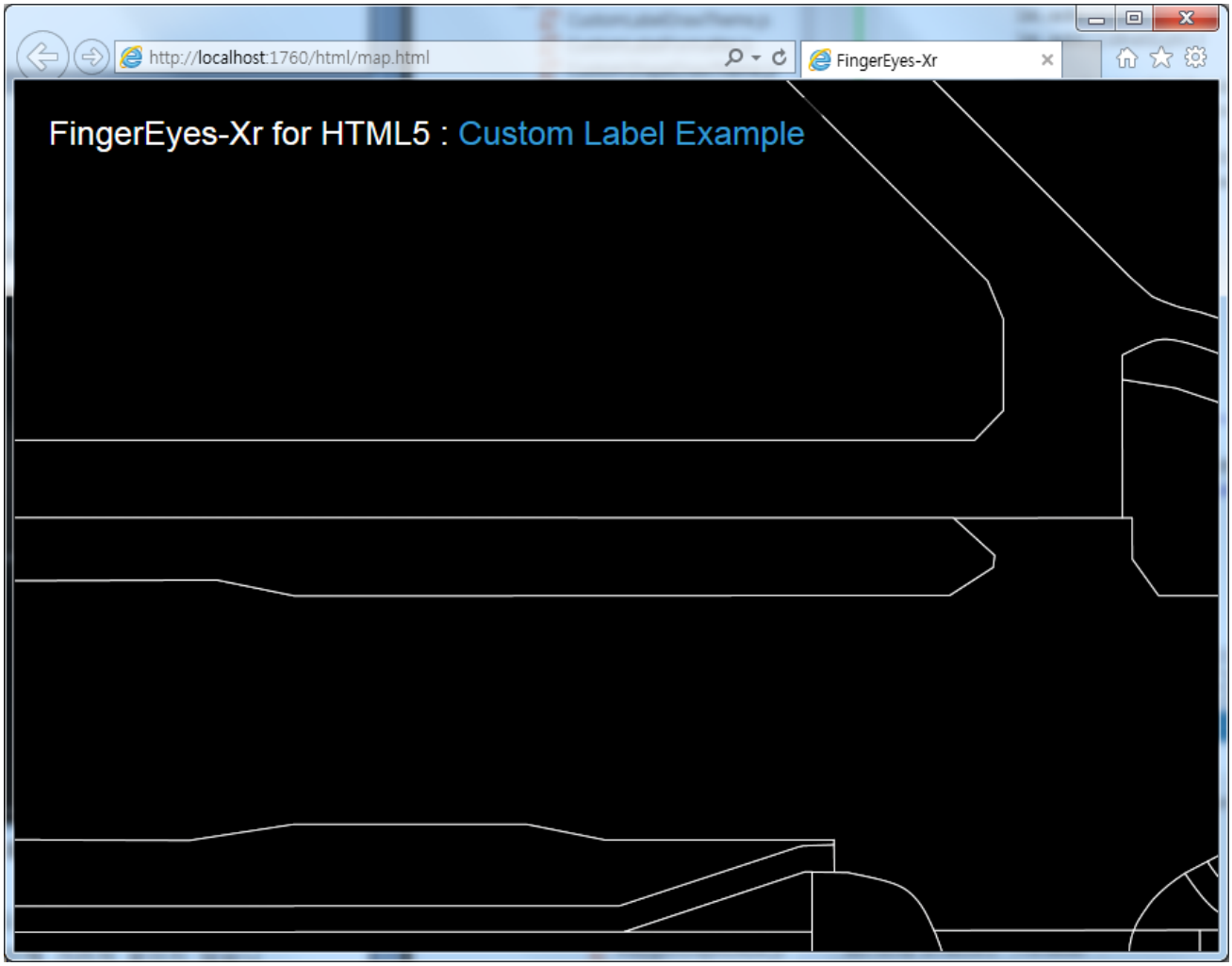


그림 5. 항공영상과 수치지도 레이어 추가에 대한 실행 화면

실행해 보면 위의 화면처럼 지적 필지가 검정색 채움과 하얀색 선으로 표시 되는 수치지도를 볼 수 있습니다. 참고로 이 수치지도 밑에는 항공 영상이 표시되고 있는데, 지적 필지로 인해 가려져 보이지 않습니다. 이제 지적 필지의 그리기 심벌을 알맞게 변경하여 항공영상도 표출되도록 하겠습니다.

먼저 아래의 코드를 **코드 9. 수치지도 레이어 생성** 부분의 수치지도 레이어 객체를 성하는 코드 바로 밑에 추가합니다.

```
01  var theme = jibunLyr.theme();
02  var pen = theme.penSymbol();
03  var brush = theme.brushSymbol();
04
05  pen.color('#ffff00');
06  pen.width(2);
07
08  brush.color('#ff0000');
09  brush.opacity(0.0);
```

코드 12. 수치지도 레이어의 그리기 심벌 설정

위의 코드를 설명하면, 먼저 1번에서 해당 수치지도 레이어의 테마(Theme)을 얻고 이 테마 객체로부터

펜(Pen)과 브러쉬(Brush) 심벌을 얻습니다. 펜은 도형의 선을 그리는데 사용하는 심벌이고 브러쉬는 도형의 채움 특성을 지정하는 심벌입니다. 펜의 색상을 노란색(#ffff00)으로 하고 굵기는 2로 합니다. 그리고 브러쉬의 투명도를 0으로 하여 완전히 투명하게 만듭니다. 실행하면 지정된 그리기 심벌로 하여 아래와 같은 화면이 나타납니다.

그림 6. 수치지도 레이어의 그리기 심벌 설정

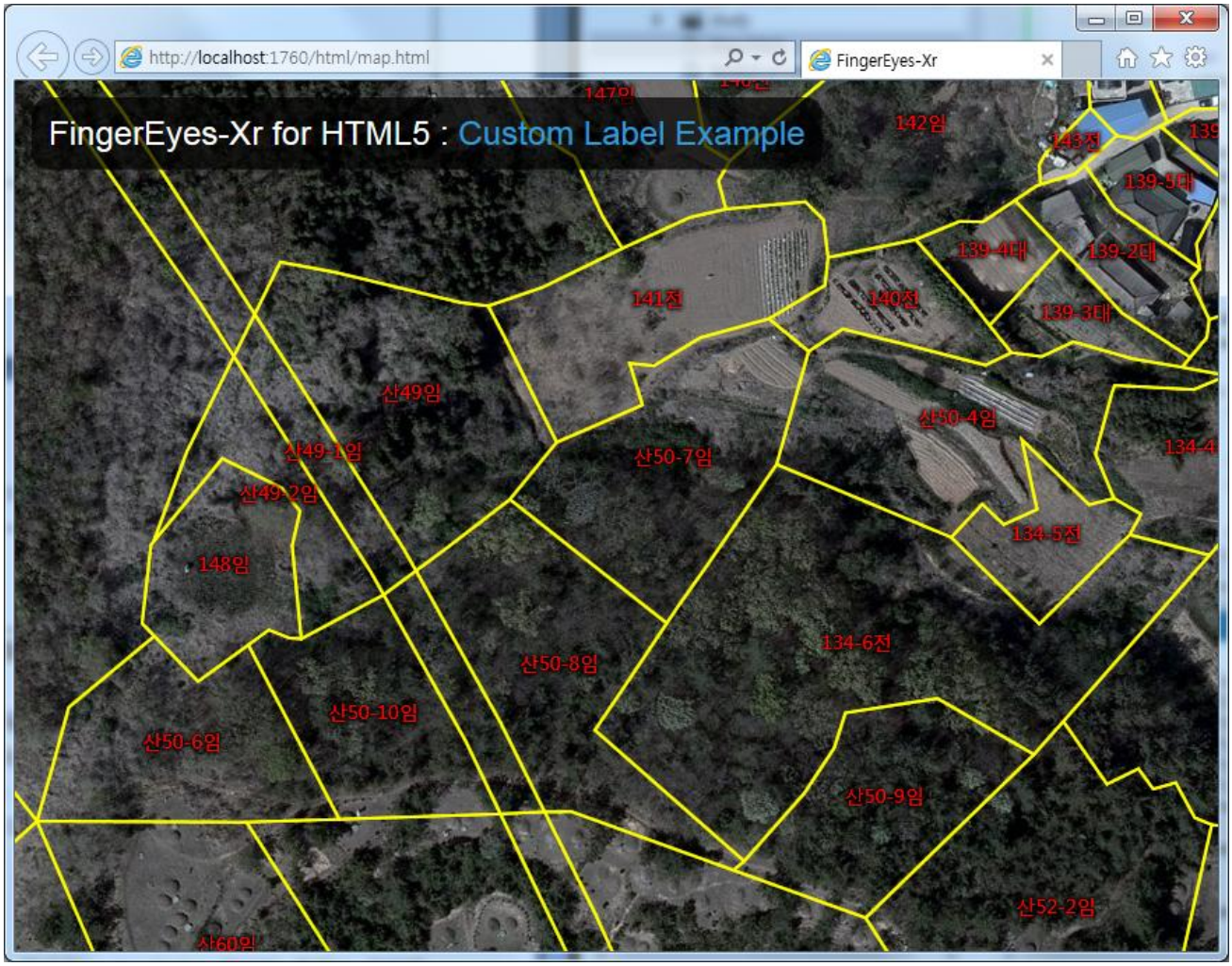
이제 라벨을 표시하겠습니다. 아래의 코드를 **코드 12. 수치지도 레이어의 그리기 심벌 설정** 코드의 마지막 부분에 추가합니다.

```
01  var label = jibunLyr.label();
02
03  label.enable(true);
04  label.formatter().fieldName("PAR_LBL");
05
06  var labelTheme = label.theme();
07  labelTheme.symbol().strokeColor("#000000");
08  labelTheme.symbol().strokeWidth(2);
```

```
09
10  labelTheme.symbol().size(14);
11  labelTheme.symbol().fontFamily('맑은 고딕');
12  labelTheme.symbol().color("#ff0000");
```

코드 13. 라벨 설정

위의 코드는 기본적인 라벨을 설정하는 코드로써, 3번은 라벨을 활성화 하는 코드이고 4번은 라벨을 위한 텍스트 값을 얻기 위해서 사용할 필드로 PAR_LBL을 사용한다는 코드입니다. 그리고 6번은 라벨 텍스트를 그리기 위한 심벌을 설정하기 위한 테마(Theme) 객체를 얻는 코드입니다. 이 테마 객체를 통해 7번 ~ 12번 코드처럼 폰트와 텍스트 색상 등을 설정합니다. 실행하면 다음과 같은 화면을 살펴볼 수 있습니다.



이제 여기서 앞서 작성한 기본적인 라벨이 아닌 지적 필지의 종류에 따라 라벨의 텍스트의 색상을 달리해 보겠습니다. 이를 위해서 **코드 13. 라벨 설정**에서 6번 ~ 12번 코드를 제거하고 아래의 코드로 대체합니다.

```
01  var labelTheme = new CustomLabelDrawTheme(jibunLyr);
02  label.theme(labelTheme);
```

위의 코드는 라벨의 텍스트를 그리는데 사용하는 심벌에 대한 테마(Theme)를 위한 객체를 생성하고 설정하는 것입니다. CustomLabelDrawTheme 클래스가 바로 라벨의 텍스트를 그리는데 사용하는 사용자 정의 테마 클래스입니다. 이 클래스에 대한 소스 코드를 CustomLabelDrawTheme.js 파일에 작성할 것입니다. 이 파일을 map.html 파일이 있는 폴더에 js 폴더를 만들고 이 js 폴더 안에 생성합니다. 그리고 다음과 같은 내용으로 코딩합니다.

```
01  CustomLabelDrawTheme = Xr.Class({
02      name: "CustomLabelDrawTheme",
03      extend: Xr.theme.ProgrammableLabelDrawTheme,
04      requires: [Xr.theme.ILabelDrawTheme],
05
06      construct: function (/* ShapeMapLayer */ layer) {
07          this.superclass(layer);
08
09          this._fieldIndex = -1;
10
11          var textSize = 12;
12
13          this._symDAB = new Xr.symbol.FontSymbol();
14          this._symDAB.strokeColor("#000000");
15          this._symDAB.strokeWidth(2);
16          this._symDAB.size(textSize);
17          this._symDAB.fontFamily('맑은 고딕');
18          this._symDAB.color("#EEB500");
19
20          this._symDO = new Xr.symbol.FontSymbol();
21          this._symDO.strokeColor("#000000");
22          this._symDO.strokeWidth(2);
23          this._symDO.size(textSize);
24          this._symDO.fontFamily('맑은 고딕');
25          this._symDO.color("#4F81BD");
26
27          this._symGONG = new Xr.symbol.FontSymbol();
28          this._symGONG.strokeColor("#000000");
29          this._symGONG.strokeWidth(2);
30          this._symGONG.size(textSize);
31          this._symGONG.fontFamily('맑은 고딕');
32          this._symGONG.color("#7F7F7F");
33
34          this._symDAI = new Xr.symbol.FontSymbol();
35          this._symDAI.strokeColor("#000000");
36          this._symDAI.strokeWidth(2);
```

```
37              this._symDAI.size(textSize);
38              this._symDAI.fontFamily('맑은 고딕');
39              this._symDAI.color("#C4BD97");
40
41              this._symJAB = new Xr.symbol.FontSymbol();
42              this._symJAB.strokeColor("#000000");
43              this._symJAB.strokeWidth(2);
44              this._symJAB.size(textSize);
45              this._symJAB.fontFamily('맑은 고딕');
46              this._symJAB.color("#7C9B3F");
47
48              this._symCHEON = new Xr.symbol.FontSymbol();
49              this._symCHEON.strokeColor("#000000");
50              this._symCHEON.strokeWidth(2);
51              this._symCHEON.size(textSize);
52              this._symCHEON.fontFamily('맑은 고딕');
53              this._symCHEON.color("#98C8E0");
54
55              this._symJEON = new Xr.symbol.FontSymbol();
56              this._symJEON.strokeColor("#000000");
57              this._symJEON.strokeWidth(2);
58              this._symJEON.size(textSize);
59              this._symJEON.fontFamily('맑은 고딕');
60              this._symJEON.color("#F8F200");
61
62              this._symIM = new Xr.symbol.FontSymbol();
63              this._symIM.strokeColor("#000000");
64              this._symIM.strokeWidth(2);
65              this._symIM.size(textSize);
66              this._symIM.fontFamily('맑은 고딕');
67              this._symIM.color("#81F21A");
68
69              this._symDefault = new Xr.symbol.FontSymbol();
70              this._symDefault.strokeColor("#000000");
71              this._symDefault.strokeWidth(2);
72              this._symDefault.size(textSize);
72              this._symDefault.fontFamily('맑은 고딕');
73              this._symDefault.color("#ffffff");
74          },
75
76          methods: {
77              /* FontSymbol */ symbol: function (/* ShapeRow */ shapeRow,
```

```
78                  /* FieldSet */ fieldSet, /* AttributeRow */ attributeRow) {
79                  if (this._fieldIndex == -1) {
80                      this._fieldIndex = fieldSet.fieldIndex("PAR_LBL");
81                  }
82
83                  var value = attributeRow.valueAsString(this._fieldIndex);
84                  if (value.indexOf("답") != -1) return this._symDAB;
85                  if (value.indexOf("도") != -1) return this._symDO;
86                  if (value.indexOf("공") != -1) return this._symGONG;
87                  if (value.indexOf("대") != -1) return this._symDAI;
88                  if (value.indexOf("잡") != -1) return this._symJAB;
89                  if (value.indexOf("천") != -1) return this._symCHEON;
90                  if (value.indexOf("전") != -1) return this._symJEON;
91                  if (value.indexOf("임") != -1) return this._symIM;
92                  return this._symDefault;
93              }
94          }
95  });
```

코드 14. CustomLabelDrawTheme 클래스를 정의

위의 소스 코드는 FingerEyes-Xr for HTML5 라이브러리에서 제공하는 JavaScript 기반의 클래스 정의를 위한 API를 사용하여 CustomLabelDrawTheme 클래스를 정의하는 것입니다. CustomLabelDrawTheme은 ILabelDrawTheme 인터페이스를 구현해야 하며, ProgrammableLabelDrawTheme 클래스를 상속 받아야 합니다. 3번 코드가 ProgrammableLabelDrawTheme 클래스를 상속받으라는 것이고 4번 코드가 ILabelDrawTheme 인터페이스를 구현하라는 것입니다. 6번의 construct 함수는 CustomLabelDrawTheme 클래스의 생성자입니다. 이 construct 함수의 7번 코드는 부모 클래스인 ProgrammableLabelDrawTheme 의 생성자를 호출하는 것입니다. 9번 코드의 this._fieldIndex는 지적 필지의 종류를 파악할 수 있는 필드명의 인덱스를 저장하기 위한 목적으로 정의한 변수입니다. 참고로 지적 필지의 종류를 파악할 수 있는 필드명은 PAR_LBL 입니다. 일단 생성자에서는 -1 값으로 초기화 해 놓고 적당한 시점에서 해당 필드의 인덱스 값을 얻어 올 것입니다. 그리고 10번 코드의 textSize 변수에 라벨의 텍스트 크기를 12로 지정하고 있고 이 변수를 이용해 라벨의 텍스트 크기를 지정할 때 사용합니다. 13번의 this._symDAB 객체는 지적필지의 '답'일 때 사용할 그리기 심벌입니다. 폰트명, 텍스트 크기, 텍스트 외곽선 색상과 굵기를 지정하고 있고 텍스트의 색상은 **그림 2. 지적 필지 종류에 따른 라벨의 텍스트 색상값**을 참고로 지정하고 있습니다. 20번 코드의 this._symDO는 지적필지가 '도'일 때 사용할 그리기 심벌이고 27번 코드의 this._symGONG은 지적필지가 '공'일 때에 대한 그리기 심벌입니다. 34번 코드의 this._symDAI는 지적필지가 '대'일 때에 대한 그리기 심벌입니다. 41번 코드의 this._symJAB은 지적필지가 '잡'일 때에 대한 그리기 심벌입니다. 48번 코드의 this._symCHEON은 지적필지가 '천'일 때에 대한 그리기 심벌입니다. 55번 코드의 this._symJEON은 지적필지가 '전'일 때에 대한 그리기 심벌입니다. 62번 코드의 this._IM은 지적 필지가 '임'일 때에 대한 그리기 심벌입니다. 심벌 객체 생성의 마지막으로 this._symDefault는 그 외의 지적필지에 대한 그리기 심벌입니다. 이제 CustomLabelDrawTheme 클래스의 매서드를 살펴 보겠습니다. CustomLabelDrawTheme 는 ILabelDrawTheme 을 구현해야 하는데, 구현해야 할 메서드는 77번 코드에서

의 symbol 입니다. symbol 매서드는 각각의 지적필지에 대한 라벨을 그릴 때 필요한 폰트 심벌을 얻어오기 위해 호출되는 함수로써 그려 질 필지의 좌표값에 대한 shapeRow와 필드 정보를 얻을 수 있는 fieldSet, 그리고 속성값을 얻을 수 있는 attributeRow를 인자로 갖습니다. 이 symbol 메서드를 살펴보면, 먼저 79번 코드에서 this._fieldIndex에 PAR_LBL 필드에 해당하는 필드 인덱스 얻어와 저장하고 있습니다. 그리고 83번에서 PAR_LBL 필드에 해당하는 값을 얻어와 value 변수에 저장하고 이 값을 이용해 지적 필지의 종류에 따라 앞서 만들어 둔 그리기 심벌을 반환하고 있습니다.

이제 이렇게 생성한 CustomLabelDrawTheme.js 파일을 map.html에서 사용할 수 있도록 **코드2. 기본 Script** 에서 2번 줄 밑에 다음 코드를 추가합니다.

```
01  <script src="js/CustomLabelDrawTheme.js"></script>
```

코드 15. CustomLabelDrawTheme.js 스크립트 추가

이제 실행해 보면 다음과 같은 결과를 볼 수 있습니다.

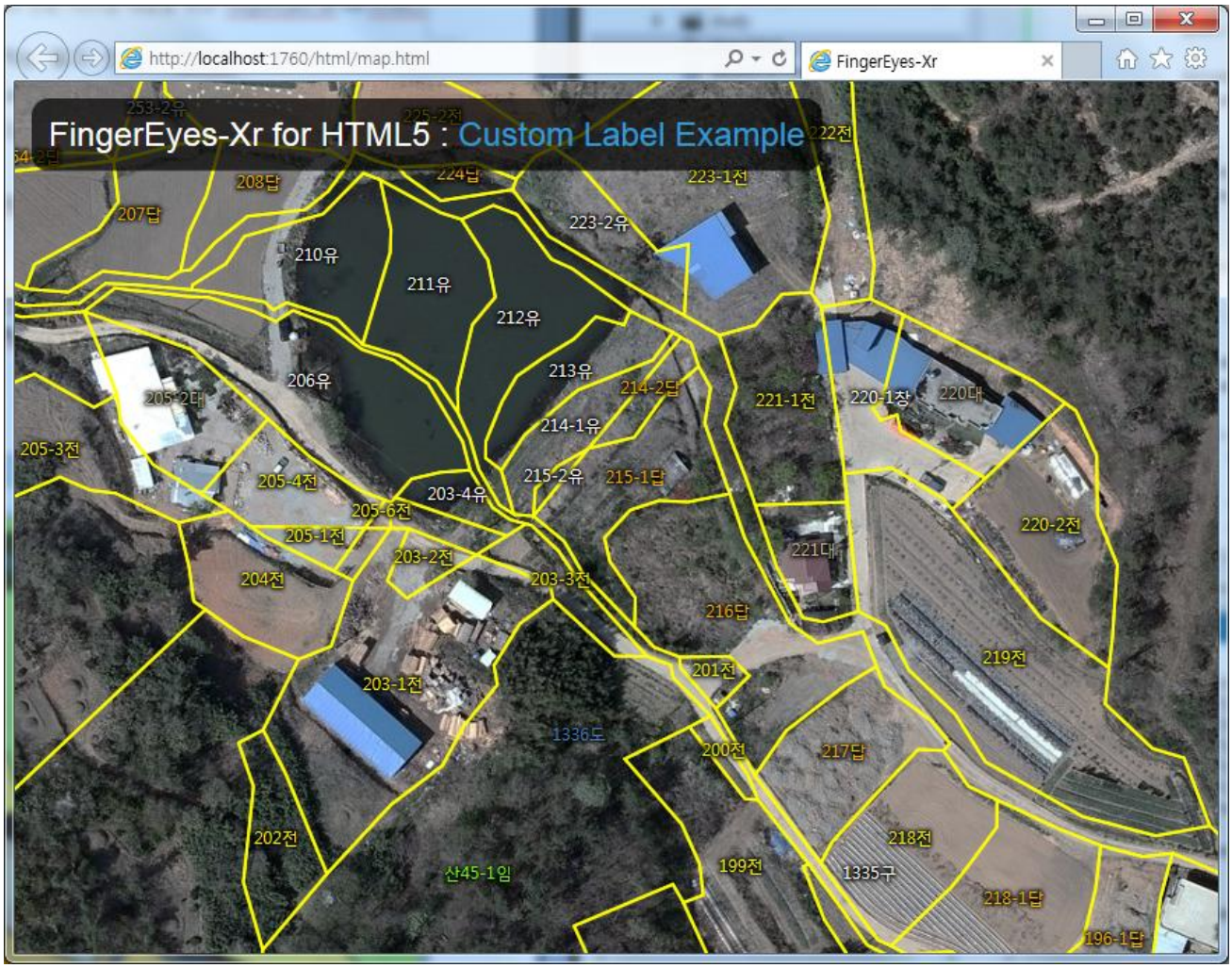


그림 7. 지적 필지 종류에 따라 다르게 표현한 지적 필지

지적 필지의 종류에 따라 지정된 텍스트 색상으로 표시되는 것을 살펴볼 수 있습니다. 라벨을 보면 221-

1전과 같이 지적 필지의 종류인 '전' 앞에 본번과 부번에 대한 번호가 붙어 있는 것을 볼 수 있습니다. 이를 번호를 제거하고 지적 필지의 종류인 '전'만을 추출해 라벨의 텍스트로 표시하는 방법에 대해 살펴보겠습니다.

코드 13. 라벨 설정에서 4번 코드를 제거하고 다음 코드를 추가합니다.

```
01  var formatter = new CustomLabelFormatter(jibunLyr);
02  label.formatter(formatter);
```

코드 16. 라벨 포매터(Formatter) 설정

CustomLabelFormatter라는 클래스의 객체를 생성하여 라벨의 포메터(Formatter)로 설정하고 있습니다. 이 CustomLabelFormatter 클래스를 생성하기 위해 CustomLabelFormatter.js 파일이 있는 동일한 폴더에 CustomLabelFormatter.js 파일을 생성하고 다음처럼 입력합니다.

```
01  CustomLabelFormatter = Xr.Class({
02      name: "CustomLabelFormatter",
03      extend: Xr.label.ProgrammableLabelFormatter,
04      requires: [Xr.label.ILabelFormatter],
05
06      construct: function (/* ShapeMapLayer */ layer) {
07          this.superclass(layer);
08          this._fieldIndex = -1;
09      },
10
11      methods: {
12          /* string */ value: function (/* ShapeRow */ shapeRow,
13              /* FieldSet */ fieldSet, /* AttributeRow */ attributeRow) {
14              if (this._fieldIndex == -1) {
15                  this._fieldIndex = fieldSet.fieldIndex("PAR_LBL");
16              }
17
18              var value = attributeRow.valueAsString(this._fieldIndex);
19
20              return value.substring(value.length-1);
21          }
22      }
23  });
```

코드 17. 라벨의 포매터(Formatter) 클래스

CustomLabelFormatter 클래스는 ILabelFormatter 인터페이스를 구현해야 하며 ProgrammableLabelFormatter 클래스를 상속받아야 합니다. CustomLabelFormatter 클래스의 생성자인 constructor 클래스를 보면 8번에서 this._fieldIndex를 -1로 초기화하고 있는데, 이 속성값은 PAR_LBL 필드에 대한 인덱스 값을 적당한 시점에서 가져와 저장하기 위해 정의 되었습니다. CustomLabelFormatter 클래스는 ILabelFormatter 인터페이스를 구현하므로 반드시 value라는 함수 정의해야 합니다. 12번 코드가 value 함수를 정의하고 있는 부분으로 이 함수를 살펴 보면, 14번에서 this._fieldIndex에 PAR_LBL 필드의 인덱스를 얻어와 저장하고 있습니다. 그리고 18번 코드에서는 PAR_LBL 필드에 해당하는 값을 가져와 value 변수(value 함수 내부의 value 변수임)에 저장하고 있습니다. 20번 코드에서 이 value 변수에 저장된 값의 끝 글자를 반환하고 있는데, 예를 들어서 '123-23대'라고 할 경우 끝 글자가 '대'이므로 '대'를 반환하게 되는 것입니다. 이렇게 반환된 문자값이 라벨에 표시되는 문자값으로 사용됩니다. 이제 이렇게 생성

한 CustomLabelFormatter.js 파일을 map.html에서 사용할 수 있도록 **코드2. 기본 Script** 에서 2번 줄 밑에 다음 코드를 추가합니다.

```
01  <script src="js/CustomLabelFormatter.js"></script>
```

코드 18. CustomLabelFormatter.js 스크립트 추가

실행하기에 앞서 라벨의 텍스트 크기를 키우기 위해 **코드 14. CustomLabelDrawTheme 클래스를 정의**에서 11번째 줄의 textSize 변수의 값을 12에서 24로 변경합니다. 이렇게 변경하고 실행하면 다음과 같은 화면을 살펴볼 수 있습니다.

그림 8. 라벨 문자열 변경

이제 지도를 조작할 수 있는 인덱스 맵, 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 19. 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.

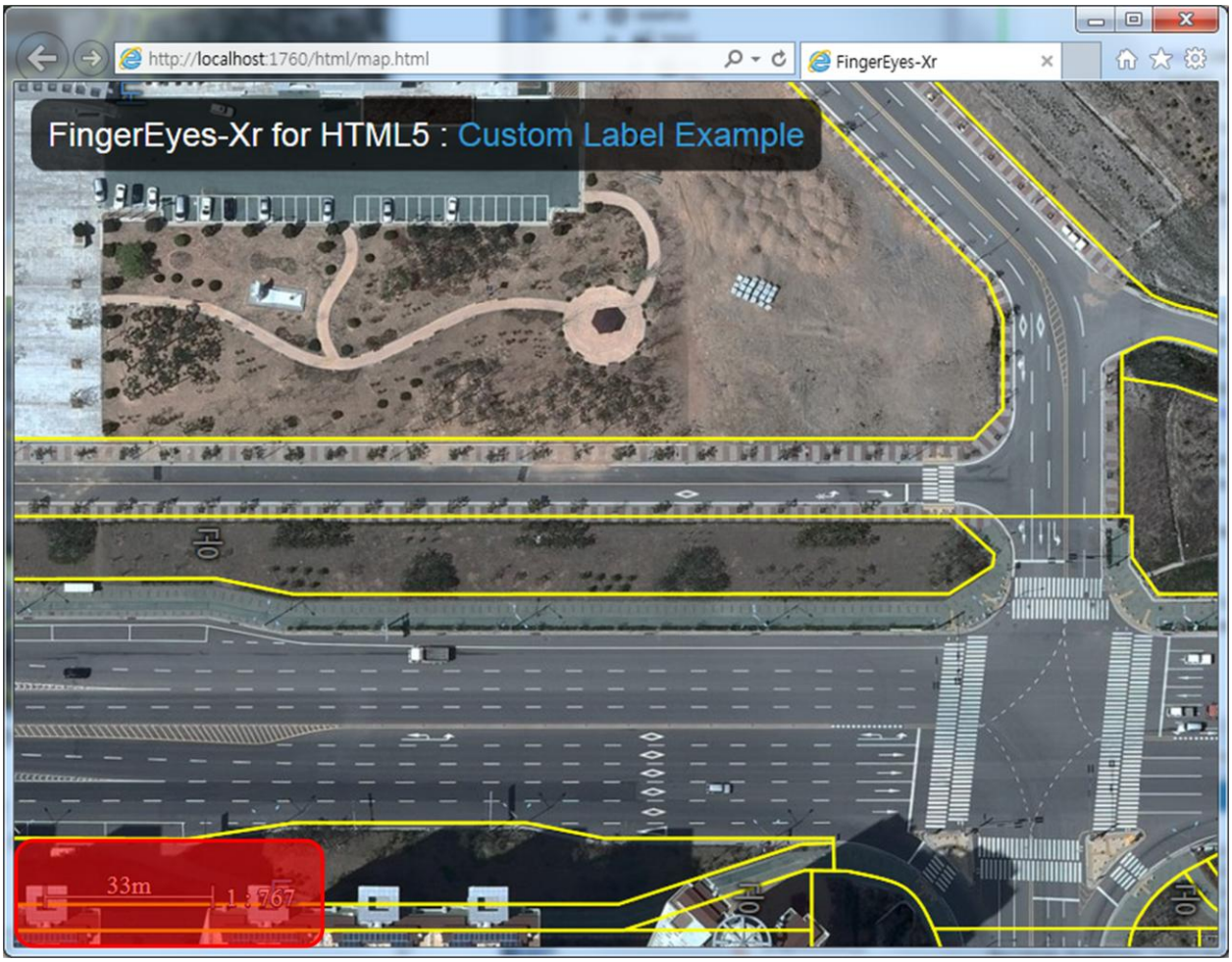


그림 9. 축척바 컨트롤 표시

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
03  ctrl2.mapScales([
04      383.06022560915073, 766.1204512466478, 1915.3011281307926,
05      3830.6022556379635, 7661.204511956243, 19153.011279550446,
06      38305.7388117497, 76612.04511893881, 153224.09023821776]);
07
08  map.userControls().add(ctrl2);
```

코드 20. 줌 레벨 컨트롤 추가

위의 코드에서 3번 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 10. 줌 레벨 컨트롤 추가

끝으로 인덱스 맵 컨트롤을 화면에 표시하도록 하겠습니다. 아래의 코드를 load 함수의 가장 마지막 줄에 추가합니다.

```
01  var ctrl3 = new Xr.ui.IndexMapControl("imc", map,
02      "http://222.237.78.208:8080/Xr?layerName=INDEXMAP");
03
04  ctrl3.size(200, 270);
05  map.userControls().add(ctrl3);
```

코드 21. 인덱스맵 컨트롤 추가

인덱스 맵 컨트롤 생성을 위해 IndexMapControl 객체를 생성하는데, 이때 생성자의 세번째 인자에 인덱스 맵에 표시되는 지도에 대한 Connection String을 지정하게 됩니다. 그리고 3번 코드에서 인덱스맵 컨트롤의 크기를 160px, 세로 300px로 지정하고 있습니다. 실행하면 다음처럼 인덱스 맵이 화면에 표시되는 것을 볼 수 있습니다.

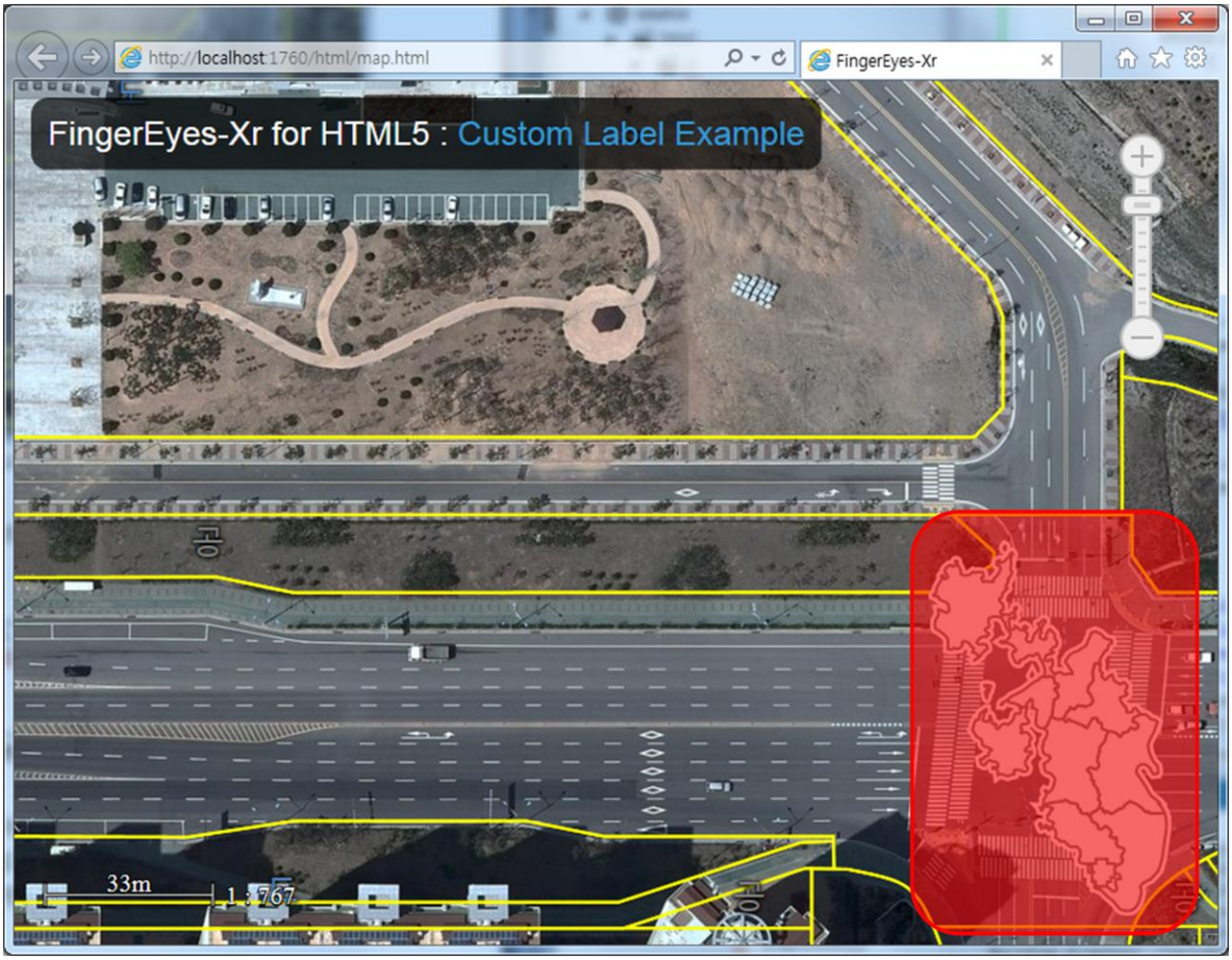


그림 11. 인덱스맵 컨트롤 표시

그런데 아직 한가지 보완해야 할 점이 있습니다. 지도를 계속 축소하다보면 아래처럼 너무나 많은 지적 필지가 그려짐으로써 필요 이상으로 복잡한 지도가 표시되고 속도가 매우 느려지게 됩니다.

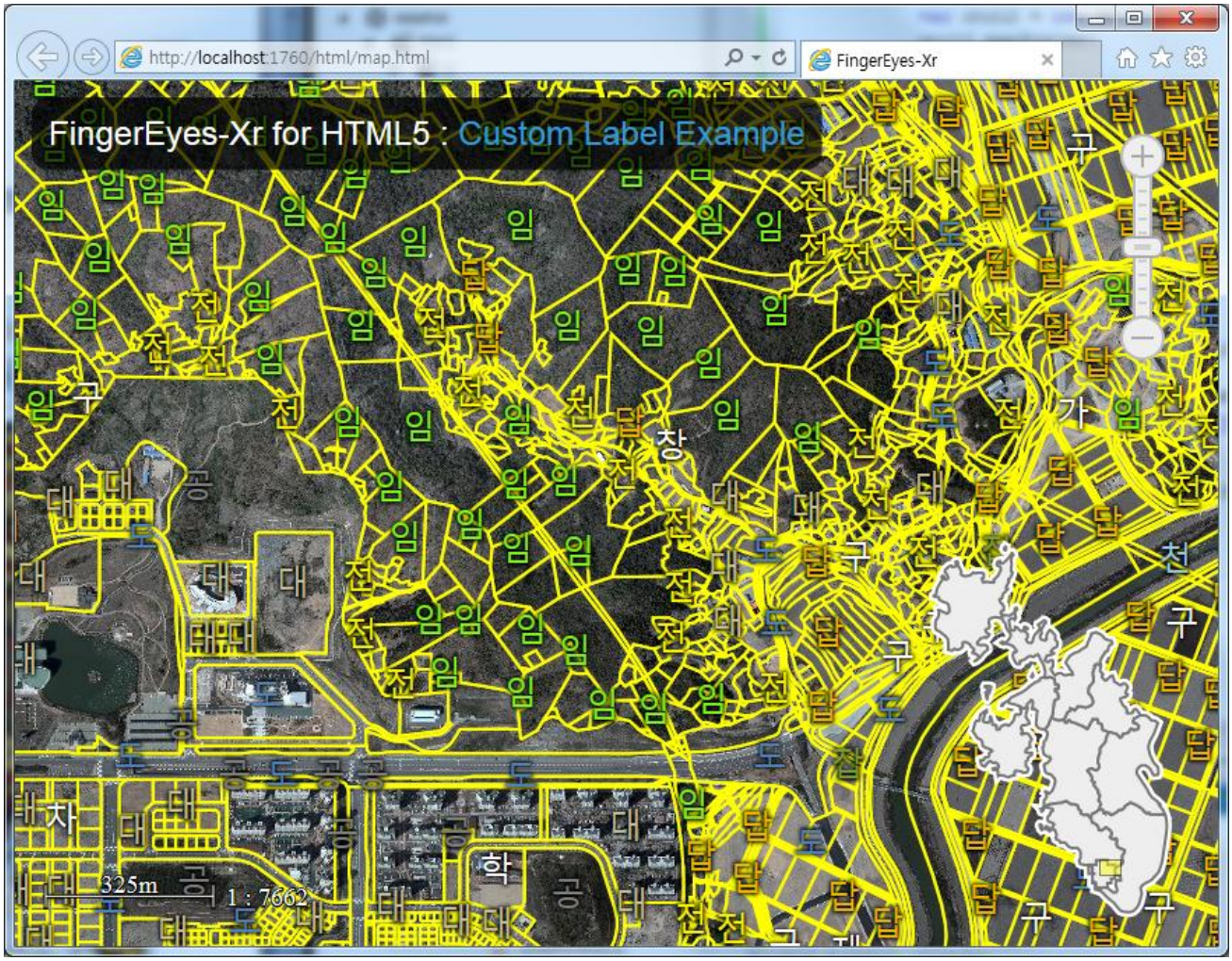


그림 12. 너무 많은 필지와 라벨이 그려져 의미없는 지도로 표현되고 속도가 느려짐

위와 같은 문제점을 개선하기 위해서는 주어진 축척 범위에서만 지적 필지와 필지의 라벨이 표시되도록 하면 됩니다 축척 범위를 1:1 ~ 1:2000로 하겠습니다. 이를 위해서 아래의 코드를 **코드 10. 생성한 레이어 추가** 부분의 가장 앞에 추가합니다.

```
01  var visibility = jibunLyr.visibility();
02  visibility.visibleByScale(true);
03  visibility.fromScale(1);
04  visibility.toScale(2001);
05
06  label.visibility().visibleByScale(true);
07  label.visibility().fromScale(1);
08  label.visibility().toScale(2001);
```

코드 22. 정해진 축척 범위에서 지적 필지 도형과 라벨 표시

범위를 지정하기 위해서는 축척의 분모값을 사용합니다. 시작 범위를 지정하기 위해서는 fromScale 함수를 사용하고 끝 범위를 지정하기 위해서는 toScale 함수를 사용합니다. 이때 fromScale로 지정한 값은 범위 조건에 포함되지만 toScale로 지정된 값은 범위 조건에 포함되지 않는다는 것에 주의하시기 바랍니다. toScale의 값을 2001로 주어 2000보다 약간 크게 잡은 이유가 바로 이런 특징 때문입니다.

이상으로 수치지도 레이어의 라벨을 그릴 때 속성값에 따라 라벨의 텍스트를 다양한 심벌로 표현하는 방법에 대한 내용과 라벨의 텍스트를 원하는 값으로 지정하는 내용을 살펴보았고, 다양한 지도 조작 컨트롤을 추가하는 내용도 살펴보았습니다. 지금까지 작성한 코드에 대한 전체 내용은 아래와 같습니다.

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

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

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

    #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: "Arial";
    }
    </style>

    <title>FingerEyes-Xr</title>

    <script src="http://www.geoservice.co.kr/z/Xr.min.1.0.js"></script>
    <script src="js/CustomLabelDrawTheme.js"></script>
    <script src="js/CustomLabelFormatter.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://222.237.78.208:8080/muan_tilemaps",
```

```
            ext: "jpg"
        }
    );

    var jibunLyr = new Xr.layers.ShapeMapLayer("jibun",
        {
            url: "http://222.237.78.208:8080/Xr?layerName=BML_PARC_AS"
        }
    );

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

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

    brush.color('#ff0000');
    brush.opacity(0.0);

    var label = jibunLyr.label();
    label.enable(true);

    var formatter = new CustomLabelFormatter(jibunLyr);
    label.formatter(formatter);

    var labelTheme = new CustomLabelDrawTheme(jibunLyr);
    label.theme(labelTheme);

    var visibility = jibunLyr.visibility();
    visibility.visibleByScale(true);
    visibility.fromScale(1);
    visibility.toScale(2001);

    label.visibility().visibleByScale(true);
    label.visibility().fromScale(1);
    label.visibility().toScale(2001);

    var lm = map.layers();

    lm.add(lyr);
    lm.add(jibunLyr);

    map.onLayersAllReady(function () {
        var cm = map.coordMapper();

        cm.moveTo(151531, 246679);
        cm.zoomByMapScale(766);

        map.update();
    });

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

    var ctrl2 = new Xr.ui.ZoomLevelControl("zlc", map);
    ctrl2.mapScales(
        [
            383.06022560915073,
            766.1204512466478,
            1915.3011281307926,
            3830.6022556379635,
            7661.204511956243,
            19153.011279550446,
```

```
                    38305.7388117497,
                    76612.04511893881,
                    153224.09023821776
                ]
                );
            map.userControls().add(ctrl2);

            var ctrl3 = new Xr.ui.IndexMapControl("imc", map,
                "http://222.237.78.208:8080/Xr?layerName=INDEXMAP");

            ctrl3.size(200, 270);
            map.userControls().add(ctrl3);

        }
    </script>
</head>
<body onload="load()">
    <div id="mainLayout">
        <div id="mapDiv"></div>
        <div id="title">
            FingerEyes-Xr for HTML5 :
            <font color="#349bd6">Custom Label Example</font>
        </div>
    </div>
</body>

</html>
```