FingerEyes-Xr for HTML5 Tutorials - 10

수치지도 레이어를 위한 마커 심볼

2015년 3월 2일, 1차 배포

㈜지오서비스

수치지도 레이어를 위한 마커 심볼

수치지도 레이어는 속성 데이터와 함께 공간 데이터를 폴리곤, 폴리라인, 포인트로 표현합니다. 이 중 포인트를 표현할 때 마커(Marker) 개념을 사용하여 포인트의 위치에 다양한 형태의 심벌로 표현할 수 있습니다. FingerEyes-Xr for HTML5에서 제공하는 포인트 타입에 대한 마커 심벌은 기본적으로 원(Circle), 사각형(Rectangle), 이미지(Image)이며 이외에도 개발자가 쉽게 더 다양한 마커 심벌을 추가 확장할 수 있습니다. 이 글에서는 포인트 타입의 수치지도에 대해 다양한 마커 심볼로 표한하는 방법에 대해 설명합니다. 참고로 이러한 마커 심벌은 수치지도의 포인트 표현 뿐만 아니라 그래픽 레이어의 포인트 표현에도 동일하게 사용될 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.

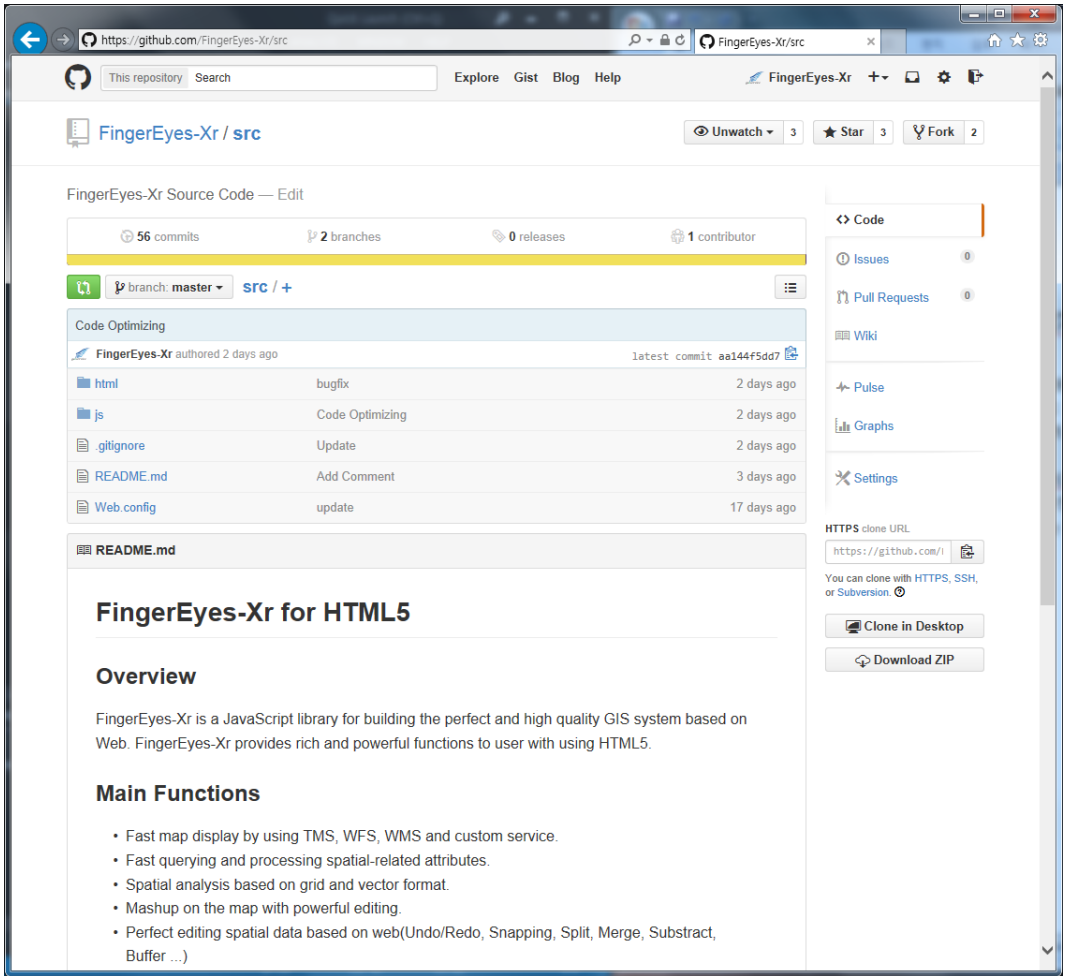

그림 1. FingerEyes-Xr for HTML에 대한 GitHub

필자는 이 글에 대한 예제 코드를 VisualStudio 2012를 이용하여 작성 하였습니다. 독자 여러분들이 어떤 툴을 사용하든 문제는 없겠으나 VisualStudio 2012에서 제공하는 JavaScript 디버깅과 기본적으로 제공하

는 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.

실습 코딩을 위해 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">Marker Symbol</font>
06             <br />
07
08             <input type="button" value="Rectangle Marker"
09                 onclick="onRectangleMarker();">
10             <input type="button" value="Circle Marker"
11                 onclick="onCircleMarker();">
12             <input type="button" value="Image Marker"
13                 onclick="onImageMarker();" />
14         </div>
15     </div>
16 </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.

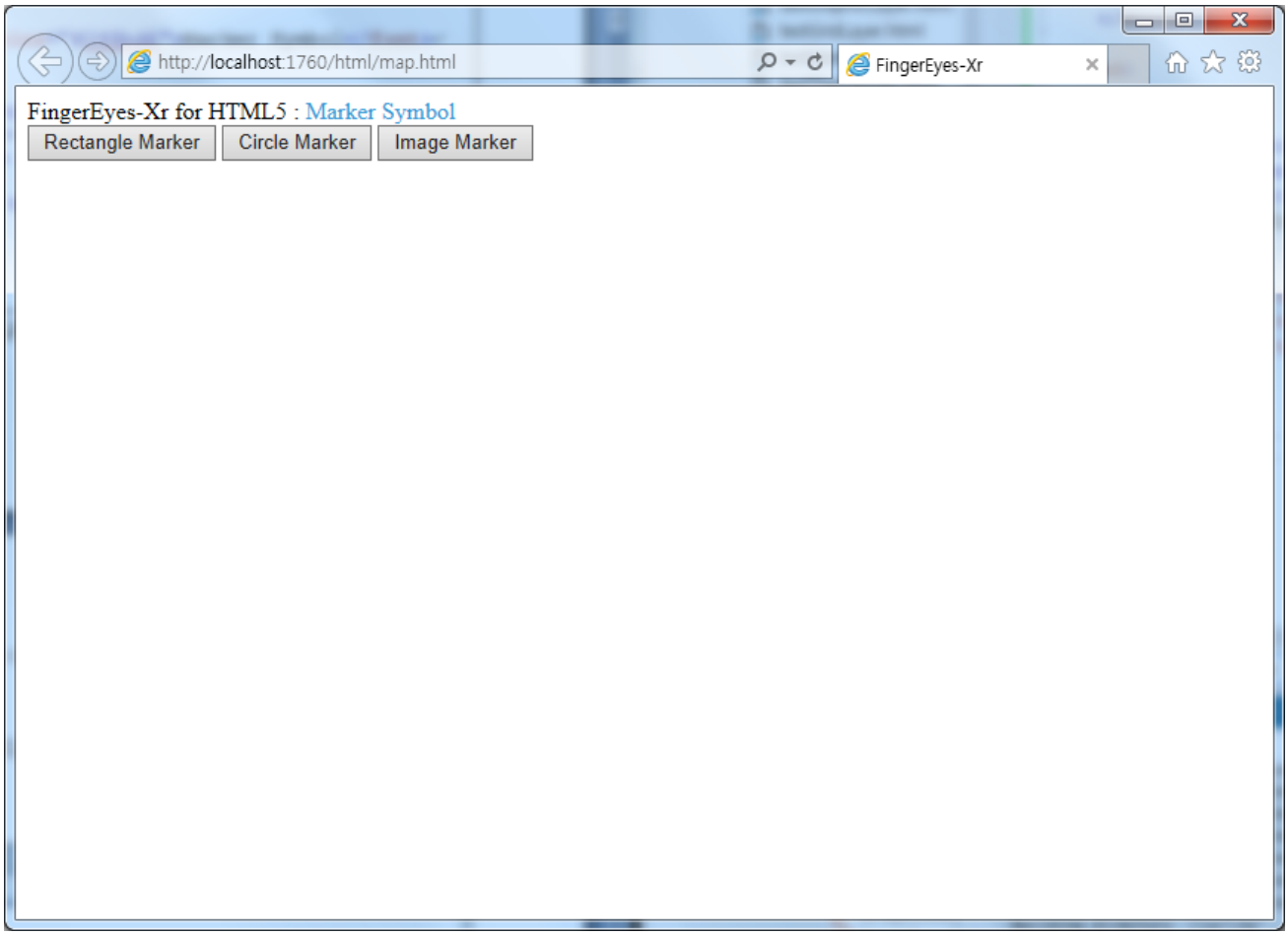

그림 2. 기본 UI 구성에 대한 실행 화면

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

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

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

    #mapDiv
    {
        top:0px;
        left:0px;
        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  </style>
41
```

코드 4. UI에 대한 스타일 적용

스타일이 적용된 페이지는 다음과 같습니다.

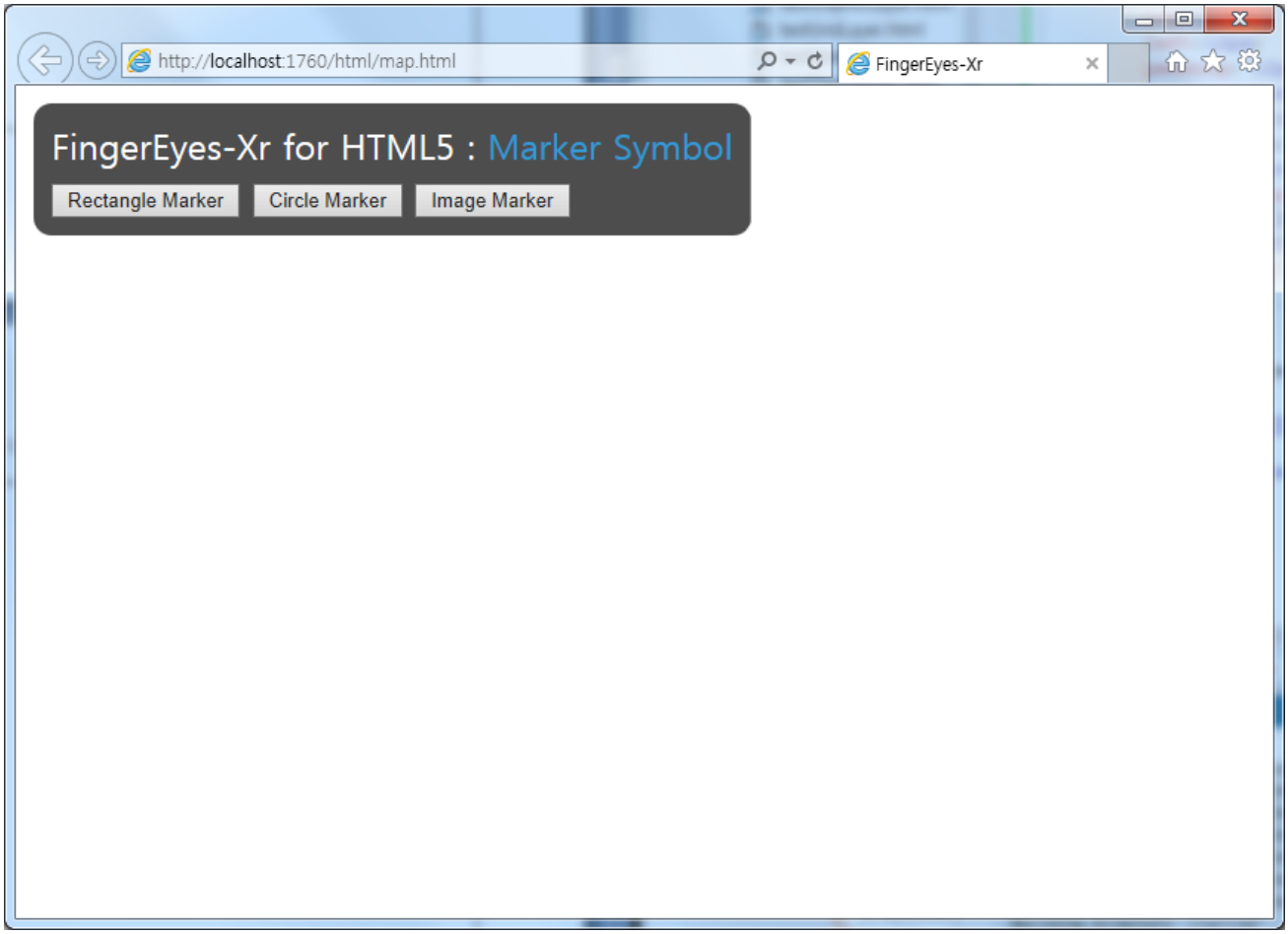

그림 3. 스타일이 적용된 UI

UI에 대해 설명하면, 3개의 버튼이 존재하며 각 버튼에 대한 기능은 아래 표와 같습니다.

버 튼	기 능 설 명
Rectangle Marker	포인트 도형을 사각형 심벌로 전환
Circle Marker	포인트 도형을 원 심벌로 전환
Image Marker	포인트 도형을 이미지 심벌로 전환

표 1. 버튼 기능 설명

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 포인트 타입의 수치지도 레이어입니다. 바로 이 포인트 타입의 수치지도 레이어에 다양한 마커 심벌을 적용할 것입니다. 먼저 배경지도로 사용할 레이어를 아래의 코드를 통해 생성하겠습니다. 아래의 코드를 load 함수의 마지막에 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 코드에서 생성한 레이어는 배경지도를 제공하는 공간서버로부터 타일맵 지도를 받아와 화면에 표시해 줄 수 있습니다.

다음으로 포인트 타입의 수치지도 레이어를 추가하겠습니다. 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" });
```

코드 9. 수치지도 레이어 생성

위의 코드는 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.

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 cm = map.coordMapper();
03      cm.moveTo(317782, 544590);
04      cm.zoomByMapScale(1534);
05
06      map.update();
07  }
```

코드 12. 레이어 추가 완료 시 호출되는 함수의 정의

위의 코드를 살펴보면, 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 3번 ~ 4번에서 화면 상에 표시될 지도의 중심 좌표와 지도의 축척 지정하고 있습니다. 최종적으로 6번 코드에서 update 함수를 호출하여 지도를 그리도록 하고 있습니다. 실행해 보면 다음과 같습니다.

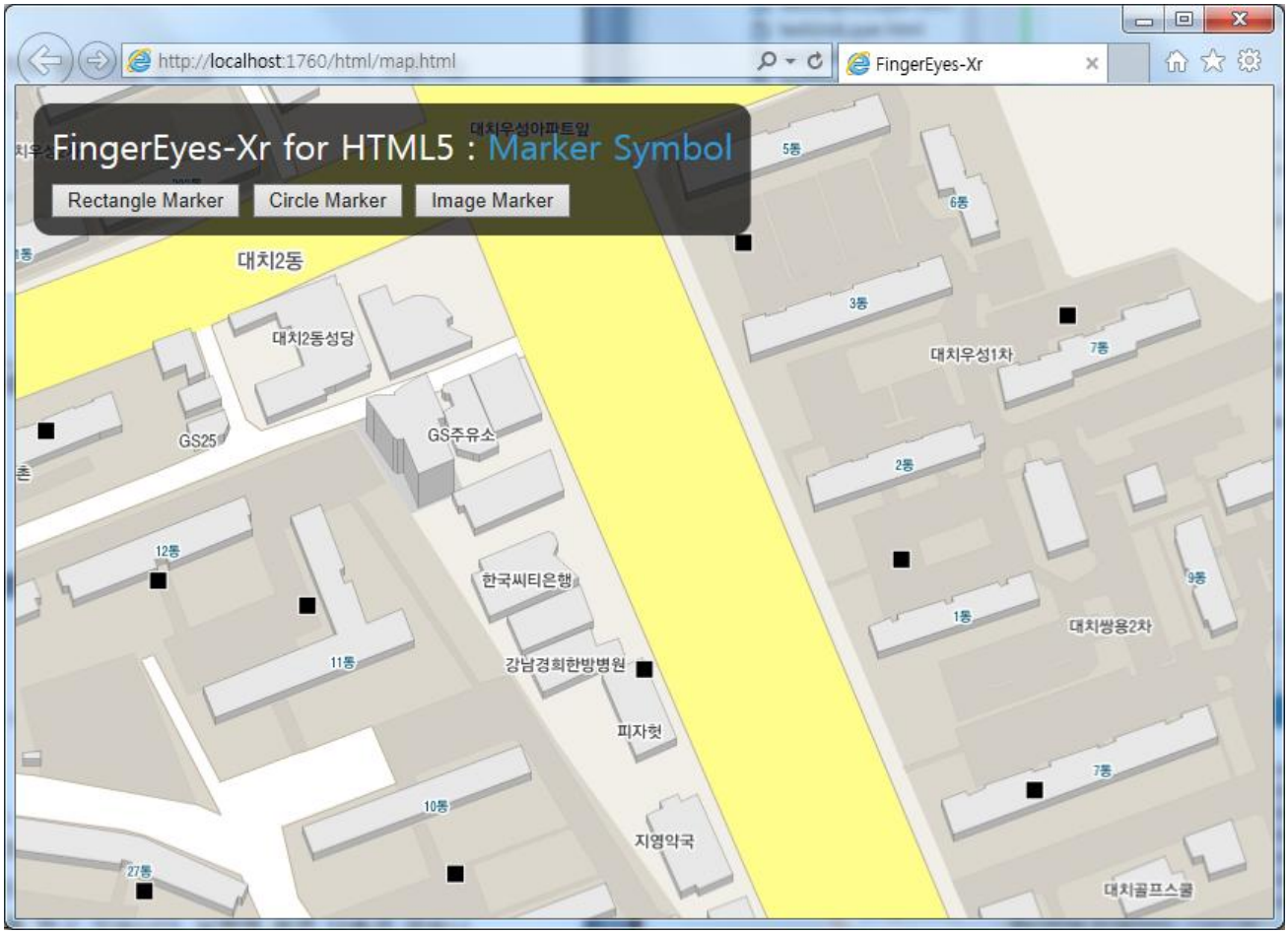

그림 4. 배경지도와 수치지도 레이어가 표시된 화면

실행해 보면 포인트 타입의 수치지도에 대한 포인트 데이터가 검정색 채움색과 하얀색 외곽선인 사각형 모양의 심벌로 표현된 것을 볼 수 있습니다. 이처럼 수치지도 레이어에서 포인트는 기본적으로 사각형 마커 심벌을 사용합니다. 다른 마커 심벌로 지정하는 것을 살펴보기 전에 기본적인 지도 관련 UI인 줌 레벨 컨트롤과 축척바 컨트롤을 추가하겠습니다.

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.

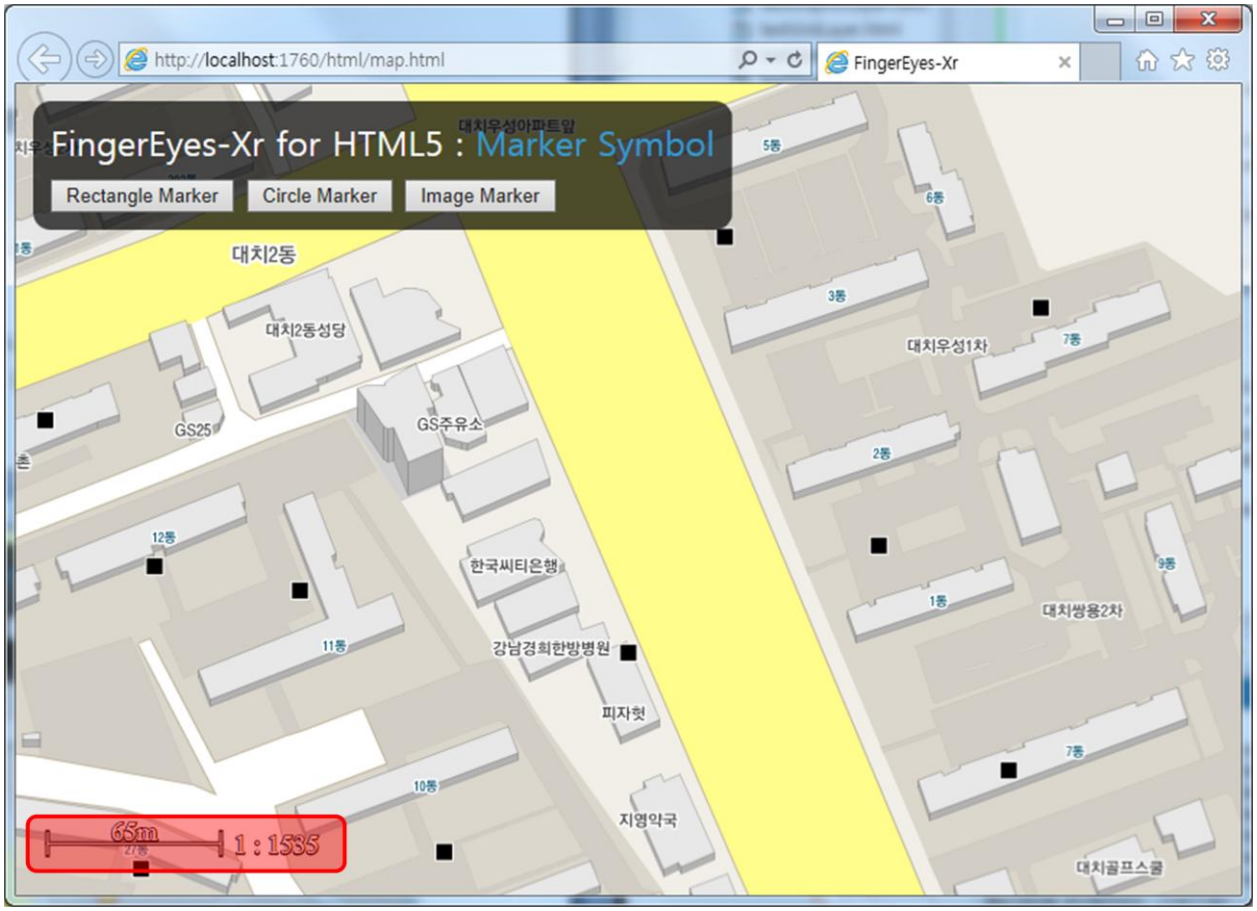

그림 5. 축척바 컨트롤 표시

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

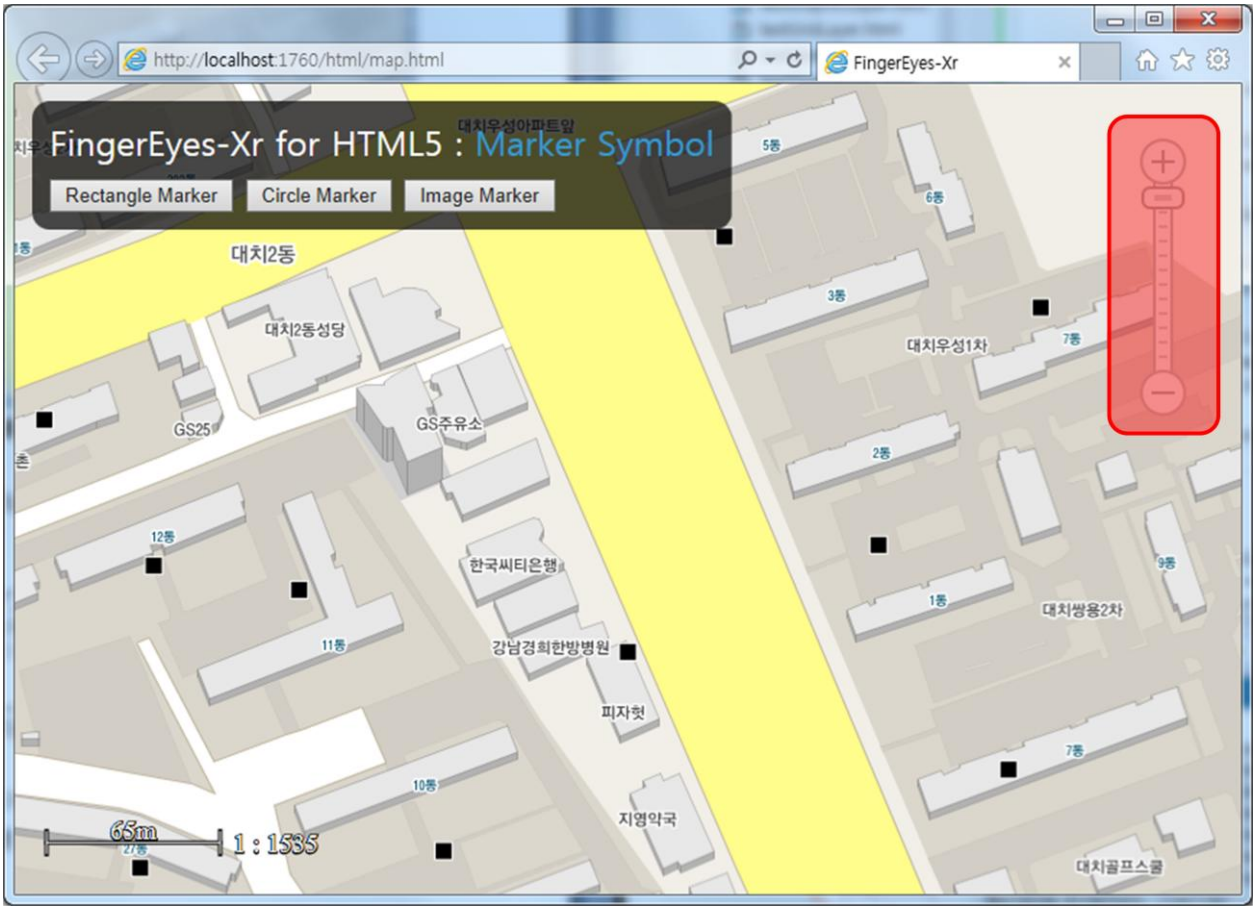

그림 6. 줌 레벨 컨트롤 추가

이제 포인트 타입의 수치지도에 대해서 다양한 마커 심벌을 적용하는 코드를 살펴 보겠습니다. UI 구성을 다시 살펴보면, 모두 3개의 버튼이 있습니다. 각 버튼에는 클릭 이벤트가 이미 지정되어 있습니다. 이 클릭 이벤트에 대한 함수를 구현하면서 다양한 마커 심벌을 적용하는 코드를 살펴보겠습니다.

먼저 사각형 마커 심벌로 변환하는 Rectangle Marker 버튼에 대한 클릭 이벤트 함수인 onRectangleMarker의 코드는 아래와 같습니다.

```
01  function onRectangleMarker() {
02      var lyr = map.layers("population");
03
04      var theme = lyr.theme();
05      var pen = theme.penSymbol();
06      var brush = theme.brushSymbol();
07
08      var markerSym = new Xr.symbol.RectangleMarkerSymbol(pen, brush);
09      markerSym.width(18).height(9);
10      theme.makerSymbol(markerSym);
11
12      pen.color('#ff0000');
13      pen.width(3);
14      brush.color('#ffff00');
15
16      map.update();
17  }
```

코드 15. 사각형 마커 심벌로 전환하는 코드

위 코드를 살펴보면, 먼저 2번 코드에서 포인트 타입의 수치지도를 레이어의 이름인 "population"으로 얻고, 4번에서 테마(Theme)를 얻습니다. 이 테마 객체에서 펜 심벌과 브러시 심벌을 얻어 각각 pen과 brush 변수에 저장해 놓습니다. 이 펜과 브러시 심벌을 사각형 마커 심벌을 표현하는데 사용됩니다. 8번 코드에서 포인트의 표현을 위한 RectangleMarkerSymbol 클래스로 사각형 마커 심벌 객체를 생성하여 markerSym 변수에 저장합니다. 9번은 사각형 마커에 대한 크기를 픽셀 단위로 지정합니다. 그리고 10번 코드에서는 수치지도 레이어에 대한 테마 객체의 markerSymbol 함수로 마커 심벌을 설정합니다. 12번 ~ 14번은 사각형 마커를 그리는데 사용하는 펜 심벌과 브러시 심벌의 함수를 이용해 색상과 선의 굵기를 지정합니다. 최종적으로 16번 코드에서 지도를 다시 그리라는 함수인 update를 호출합니다. 이제 실행하고 Rectangle Marker 버튼을 클릭해 보면 다음처럼 심벌이 변경되는 것을 볼 수 있습니다.

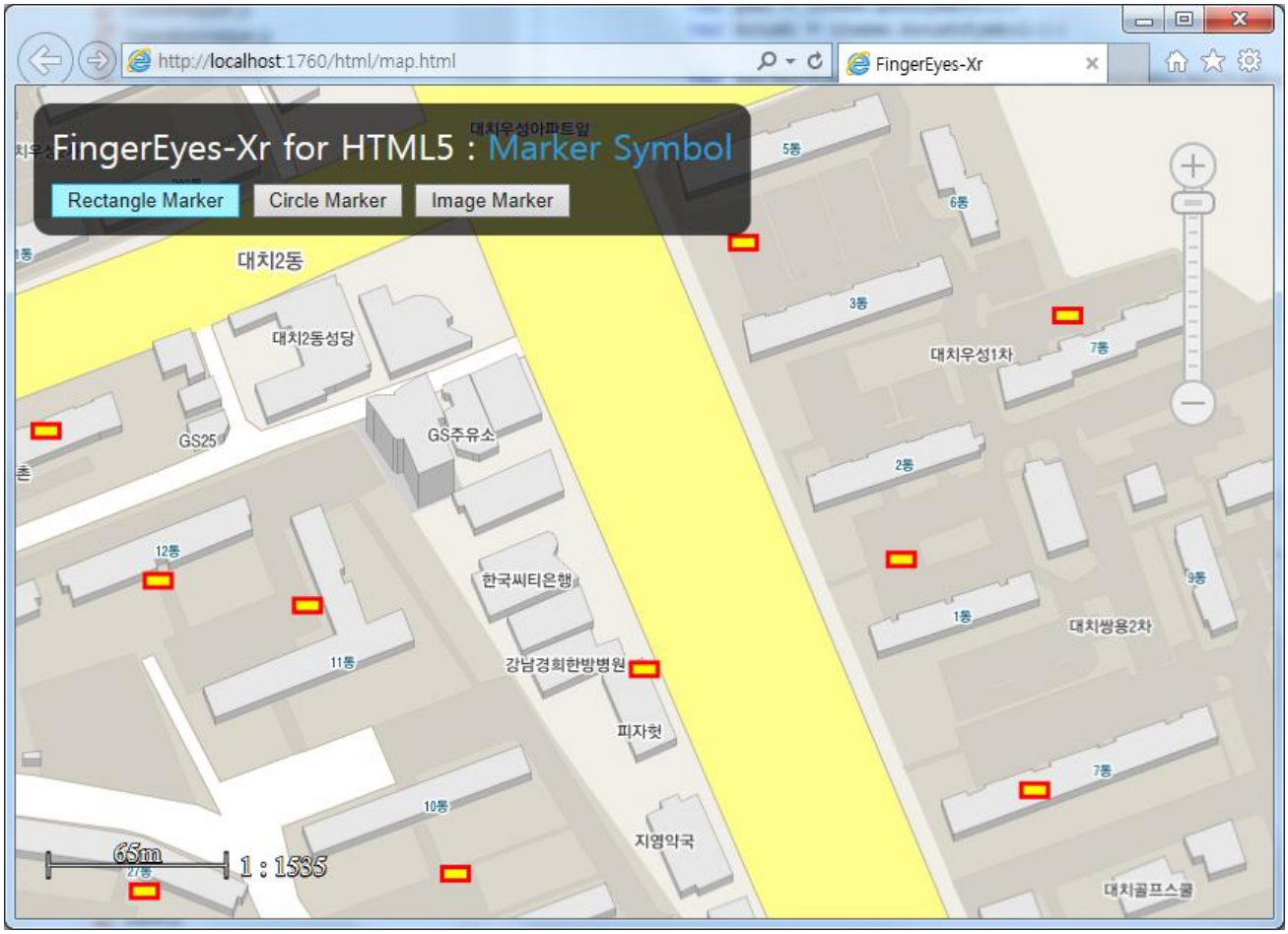

그림 7. 사각형 마커 심벌 전환

다음으로 Circle Marker 버튼에 대한 클릭 이벤트에 지정된 함수인 onCircleMarker 함수에 대해 살펴보겠습니다. 이 버튼은 포인트에 대한 마커 심벌을 원으로 표현합니다. onCircleMarker 함수는 아래와 같습니다.

```
01  function onCircleMarker() {
02      var lyr = map.layers("population");
03
04      var theme = lyr.theme();
05      var pen = theme.penSymbol();
06      var brush = theme.brushSymbol();
```

```
07
08      var markerSym = new Xr.symbol.CircleMarkerSymbol(pen, brush);
09      markerSym.radius(9);
10      theme.makerSymbol(markerSym);
11
12      pen.color('#0000ff');
13      pen.width(3);
14      brush.color('#ff0000');
15
16      map.update();
17  }
```

코드 16. 원 마커 심벌로 전환하는 코드

위 코드를 살펴보면, 먼저 2번 코드에서 포인트 타입의 수치지도를 레이어의 이름인 "population"으로 얻고, 4번에서 테마(Theme)를 얻습니다. 이 테마 객체에서 펜 심벌과 브러시 심벌을 얻어 각각 pen과 brush 변수에 저장해 놓습니다. 이 펜과 브러시 심벌을 원 마커 심벌을 표현하는데 사용됩니다. 8번 코드에서 포인트의 표현을 위한 CircleMarkerSymbol 클래스로 원 마커 심벌 객체를 생성하여 markerSym 변수에 저장합니다. 9번은 원 마커에 대한 반지름을 픽셀 단위로 지정합니다. 그리고 10번 코드에서는 수치지도 레이어에 대한 테마 객체의 markerSymbol 함수로 마커 심벌을 설정합니다. 12번 ~ 14번은 원 마커를 그리는데 사용하는 펜 심벌과 브러시 심벌의 함수를 이용해 색상과 선의 굵기를 지정합니다. 최종적으로 16번 코드에서 지도를 다시 그리라는 함수인 update를 호출합니다. 이제 실행하고 Circle Marker 버튼을 클릭해 보면 다음처럼 심벌이 변경되는 것을 볼 수 있습니다.

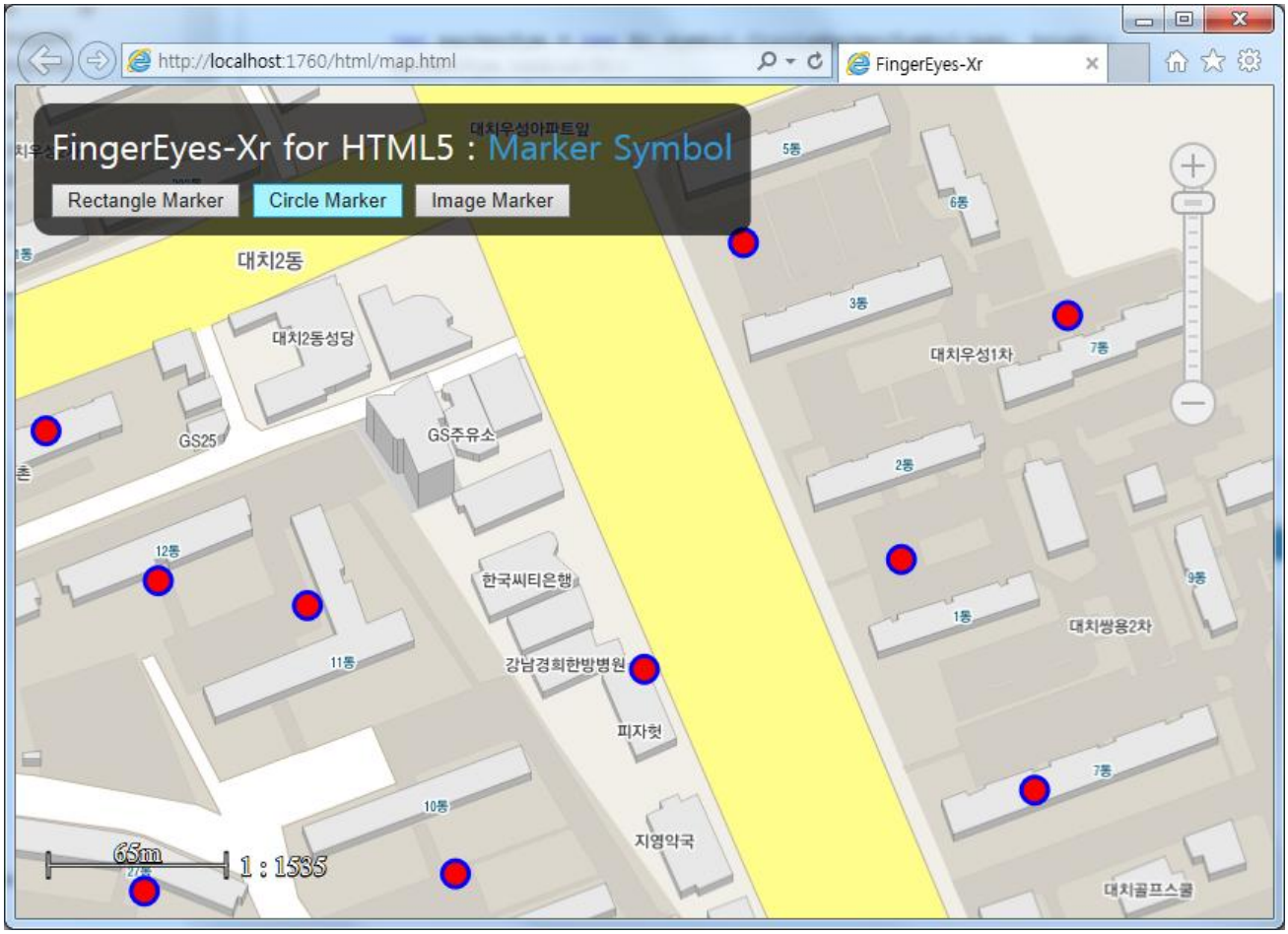

그림 8. 원 마커 심벌로 전환

다음으로 이미지 마커 심벌로 변경하는 Image Marker 버튼의 클릭 이벤트에 대한 함수인 onImageMarker를 살펴보겠습니다. onImageMarker 함수는 다음과 같습니다.

```
01  function onImageMarker() {
02      var lyr = map.layers("population");
03      var theme = lyr.theme();
04      var markerSym = new Xr.symbol.ImageMarkerSymbol();
05
06      markerSym.url("http://www.geoservice.co.kr/house_icon.png");
07      markerSym.width(64).height(64);
08
09      theme.makerSymbol(markerSym);
10
11      map.update();
12  }
```

코드 17. 이미지 마커 심벌로 전환하는 코드

위 코드를 살펴보면, 먼저 2번 코드에서 포인트 타입의 수치지도를 레이어의 이름인 "population"으로 얻고, 3번에서 테마(Theme)를 얻습니다. 4번 코드에서 이미지의 표현을 위한 ImageMarkerSymbol 클래스로 이미지 마커 심벌 객체를 생성하여 markerSym 변수에 저장합니다. 6번은 이미지 마커에서 사용할 이미지 자원에 대한 URL을 지정합니다. 7번은 이미지의 가로와 세로 크기를 64 픽셀로 지정하고 있습니다. 그리고 9번 코드에서는 수치지도 레이어에 대한 테마 객체의 markerSymbol 함수로 마커 심벌을 설정합니다. 최종적으로 11번 코드에서 지도를 다시 그리라는 함수인 update를 호출합니다. 이제 실행하고 Image Marker 버튼을 클릭해 보면 다음처럼 심벌이 변경되는 것을 볼 수 있습니다.

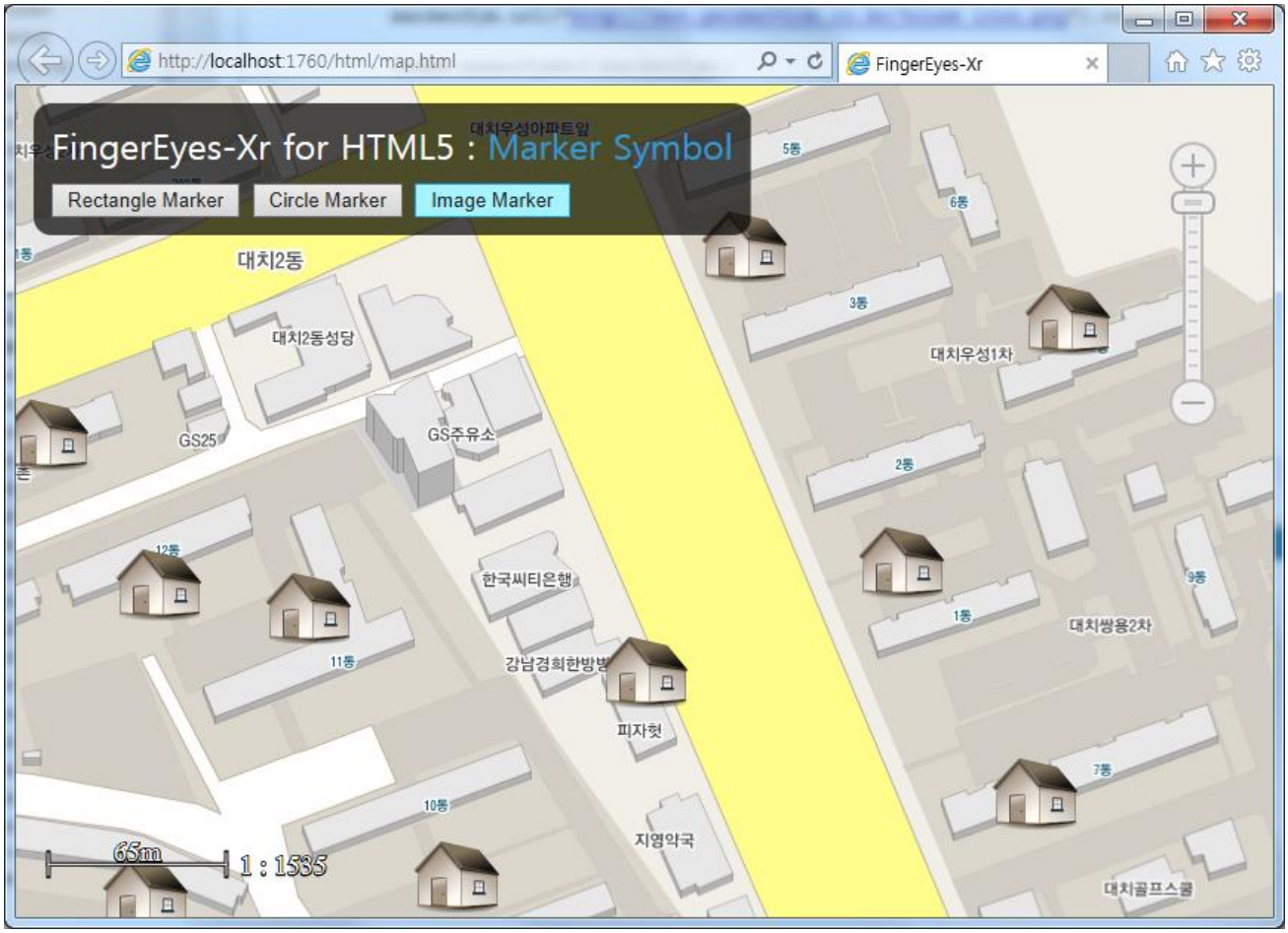

그림 9. 이미지 마커 심벌로 전환

이상으로 이미지 타입의 도형에 대해 지정할 수 있는 마커 심벌로써 사각형, 원, 이미지에 대해 살펴보았습니다. FingerEyes-Xr for HTML5에서는 기본적으로 이 세가지 형태의 마커를 지원하지만 이외에도 더 다양한 형태의 마커 심벌을 추가 확장할 수 있습니다. 아래는 지금까지 작성한 코드에 대한 전체 내용입니다.

```
<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";
    }
    </style>

    <title>FingerEyes-Xr</title>

    <!--
    <script src="http://www.geoservice.co.kr/z/Xr.min.1.0.js"></script>
    -->

    <script src="../js/Xr.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" });


            var lm = map.layers();

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

            map.onLayersAllReady(onLayersAllReady);

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

            map.update();
        }

        function onRectangleMarker() {
            var lyr = map.layers("population");

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

            var markerSym = new Xr.symbol.RectangleMarkerSymbol(pen, brush);
            markerSym.width(18).height(9);
            theme.makerSymbol(markerSym);

            pen.color('#ff0000');
            pen.width(3);
            brush.color('#ffff00');

            map.update();
        }

        function onCircleMarker() {
            var lyr = map.layers("population");

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

            var markerSym = new Xr.symbol.CircleMarkerSymbol(pen, brush);
            markerSym.radius(9);
            theme.makerSymbol(markerSym);

            pen.color('#0000ff');
            pen.width(3);
            brush.color('#ff0000');

            map.update();
        }

        function onImageMarker() {
            var lyr = map.layers("population");

            var theme = lyr.theme();

            var markerSym = new Xr.symbol.ImageMarkerSymbol();
            markerSym.url("http://www.geoservice.co.kr/house_icon.png");
            markerSym.width(64).height(64);

            theme.makerSymbol(markerSym);

            map.update();
        }

    </script>
```

```
</head>

<body onload="load()">
    <div id="mainLayout">
        <div id="mapDiv"></div>
        <div id="title">
            FingerEyes-Xr for HTML5 : <font color="#349bd6">Marker Symbol</font>
            <br />

            <input type="button" value="Rectangle Marker"
                onclick="onRectangleMarker();">

            <input type="button" value="Circle Marker"
                onclick="onCircleMarker();">

            <input type="button" value="Image Marker"
                onclick="onImageMarker();" />
        </div>
    </div>
</body>

</html>
```