Contents

Bing Maps AJAX Control, Version 7.0

Bing Maps™ is an online mapping service that enables users to search, discover, explore, plan, and share information about specific locations. By using enhanced road maps, labeled aerial photo views, and low-angle high-resolution aerial photos, Bing Maps AJAX Control 7.0, in conjunction with the Bing Maps REST Services, provides unique opportunities for developers to incorporate both location and local search features into their Web applications.

The Bing Maps AJAX Control 7.0 software development kit (SDK) consists of a complete set of reference topics that cover the Bing Maps AJAX Control 7.0 application programming interface (API).

If you are reading this help file online, you can download either the CHM or PDF version of this SDK for offline viewing.

In This Section

- Getting Started with the 7.0 Map Control
- What's New in the AJAX Map Control
- Developing with the 7.0 Map Control
- API Reference
- Supported Browsers
- Developer Resources

See Also

Terms and Conditions

Getting Started with the 7.0 Map Control

The Bing Maps AJAX Control 7.0 is a JavaScript control that contains the objects, methods, and events that allow you to display maps powered by Bing Maps on your Web site. The sections in this topic describe the steps you need to take to start using the Bing Maps AJAX Control 7.0.

Create a Bing Maps Account and Get a Key

Before you begin developing your application, you need to create a developer account on the Bing Maps Account Center. A Bing Maps Developer Account allows you to create a Bing Maps Key to use in your map application. Getting a key is described in Getting a Bing Maps Key.

When the Bing Maps AJAX Control 7.0 is loaded with a valid Bing Maps Key, Bing Maps counts sessions. A session begins with the load of the Bing Maps AJAX Control 7.0 into a user's browser and includes all Bing Maps AJAX Control 7.0 interactions until the browser is closed or the user moves to a different page. Detailed information about Bing Maps usage reports is found in Viewing Bing Maps Usage Reports.

Get Familiar with the Bing Maps AJAX 7.0 Control

The Developing with the 7.0 Map Control section of this SDK contains topics that describe how to use the features provided by the AJAX map control.

What's New in the AJAX Map Control

Welcome to the latest release of the Bing Maps AJAX Control 7.0! This is an overview of the new features in this release.

New Features

This release of the map control includes the following new features:

- **Built-in info box functionality.** Using the Infobox Class and the InfoboxOptions Object, you can easily put info boxes on the map as well as create and customize your own pushpin balloons.
- **Support for keyboard events.** With the addition of the Map events *keyup*, *keydown*, and *keypress* and the KeyEventArgs Object, you can customize keyboard event behavior in your application.
- **New mouse event argument properties.** The new *wheelDelta*, *isPrimary*, *isSecondary*, and *isTouchEvent* properties of the MouseEventArgs Object allow you to more easily handle mouse and touch events.
- **New map options.** For increased flexibility, several new options have been added to the MapOptions Object, including the *showMapTypeSelector* which allows you to hide the map type selector in the map navigation control.
- **Additional mobile and desktop browser support.** RIM BlackBerry 6.0 and Firefox 4.0 are now supported browsers. See Bing Maps AJAX Control 7.0 Supported Browsers for more information.

Developing with the 7.0 Map Control

The topics in this section will help you to start using the Bing Maps AJAX Control 7.0.

In This Section

Loading the AJAX Map Control

This topic describes how to load the Bing Maps AJAX Control 7.0 into your Web page to display a map. This is the first step you need to take for any page that uses the map control.

Displaying the Default Map

Displaying the default map, which includes all of the navigation functionality, consists of the following steps:

1. At the top of the HTML page add the following DOCTYPE declaration.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

2. In the header section of an HTML page, add a META element with the **charset** attribute set to **"utf-8"**, as follows.

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-
8">
```

   It is recommended that you use UTF-8 encoding in your web page.

3. Also in the header section, add a reference to the map control, as follows.

```
<script charset="UTF-8" type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?
v=7.0">
</script>
```

   To use SSL, add the *s* parameter to the reference as shown below.

```
<script charset="UTF-8" type="text/javascript"
src="https://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx
?v=7.0&s=1">
</script>
```

4. In the body of the page, add a DIV element to the page to contain the map. The size of the map is defined by the height and width of the DIV element. The position of the map is set by using the "position", "top", and "left" properties. You can set these values either inline or by defining the values in a style class and then referencing that class, as follows.

   ```
   <div id='mapDiv' style="position:absolute; width:400px;
   height:400px;"></div>
   ```

   or

   ```
   .map {
      position: absolute;
      top: 20;
      left: 10;
      width: 400px;
      height: 400px;
      border:#555555 2px solid;
   }
   ...
   <div id="mapDiv" class="map"></div>
   ```

   If you do not specify a width/height, the width/height of the div is used. For cross-browser compatibility, you should always specify the position attribute (both "absolute" and "relative" are valid values). If you use a percentage width and or height in the map DIV, it is the percentage of the parent width or height, respectively.

5. Finally, create a new instance of the Map Class. You also need to include a map options object to contain your credentials, which is your Bing Maps Key. See the Getting a Bing Maps Key topic.

   ```
   var map = new
   Microsoft.Maps.Map(document.getElementById("mapDiv"),
   {credentials:"Your Bing Maps Key"});
   ```

Customizing the Map When Loading

You can also specify other options when the map is first loaded, such as the location, zoom level, and the imagery of the map. To do this, pass in MapOptions or ViewOptions to the Map constructor. The code below sets the imagery to Aerial.

```
var mapOptions = {

credentials: "Your Bing Maps Key",

mapTypeId: Microsoft.Maps.MapTypeId.aerial

}


var map = new Microsoft.Maps.Map(document.getElementById("mapDiv"), mapOptions);
```

Example

The following code shows a complete Web page that loads a map. Valid map types are found in the MapTypeId Enumeration topic.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

   <head>

      <title></title>

      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">


      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>


      <script type="text/javascript">

      function GetMap()

      {


         var map = new Microsoft.Maps.Map(document.getElementById("mapDiv"),

                           {credentials: "Your Bing Maps Key",

                            center: new Microsoft.Maps.Location(45.5, -122.5),

                            mapTypeId: Microsoft.Maps.MapTypeId.road,

                            zoom: 7});

      }

      </script>

   </head>

   <body onload="GetMap();">

      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>
   </body>
</html>
```

Changing the Map View

This topic describes how to change the map that is displayed.

Setting the Initial Map View

You can set the map view when you first load the map you can use any of the options available in the MapOptions Object or ViewOptions Object.

The code below initializes the map with a specific view. The imagery displayed is set to Bird's eye using the `mapTypeId` option. Valid map type IDs are listed in the MapTypeId Enumeration topic.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">
      function GetMap()
      {
         var mapOptions = {
         credentials: "Your Bing Maps Key",
         center: new Microsoft.Maps.Location(47.592, -122.332),
         mapTypeId: Microsoft.Maps.MapTypeId.birdseye,
         zoom: 17,
         showScalebar: false
         }

         var map = new Microsoft.Maps.Map(document.getElementById("mapDiv"), mapOptions);
      }

      </script>

   </head>

   <body onload="GetMap();">

      <div id='mapDiv' style="position:relative; width:600px; height:600px;"></div>

   </body>

</html>
```

Changing the Map View

If you want to change the map after it has loaded, use the `setView` method of the Map Class. The ViewOptions Object contains available options that can be set.

The example below sets the map view to the specified zoom level.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

   <head>

      <title></title>

      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">


      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>


      <script type="text/javascript">


      var map = null;


      function GetMap()

      {

         map = new Microsoft.Maps.Map(document.getElementById("mapDiv"), {credentials:
"Your Bing Maps Key", mapTypeId: Microsoft.Maps.MapTypeId.road});

      }


      function SetZoom()

      {
         var zoomLevel = parseInt(document.getElementById('txtZoom').value);

         map.setView({zoom:zoomLevel});

      }


      </script>

   </head>

   <body onload="GetMap();">

      <div id='mapDiv' style="position:relative; width:500px; height:500px;"></div>

      Zoom Level:

      <input id="txtZoom" type="text" value="1"/>

      <input id="btnZoom" type="button" value="Click to set the zoom level"
onclick="SetZoom();"/>


   </body>

</html>
```

To set the boundaries of the view instead of centering on a point, use the `bounds` option as shown in the code below.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

   <head>

      <title></title>

      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">


      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>


      <script type="text/javascript">


      var map = null;


      function GetMap()

      {
         map = new Microsoft.Maps.Map(document.getElementById("mapDiv"), {credentials:
"Your Bing Maps Key"});

         var viewBoundaries = Microsoft.Maps.LocationRect.fromLocations(new
Microsoft.Maps.Location(47.618594, -122.347618), new Microsoft.Maps.Location(47.620700, -
122.347584), new Microsoft.Maps.Location(47.622052, -122.345869));

         map.setView({ bounds: viewBoundaries});

      }

      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative; width:500px; height:500px;"></div>
   </body>
</html>
```

Adding Entities to the Map

This topic describes how to add entities to the map. An Entity can be any one of the following types: Polygon, Polyline, Pushpin, TileLayer, or EntityCollection. Information about working with tile layers is in the Working with Tile Layers topic.

Adding Single Entities to the Map

To add a pushpin, polygon, or polyline to the map, simply create your object then add the entity to the entities property of the map.

Adding a Pushpin

The following code adds a pushpin to the center of the map with the label “1”.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         function GetMap()
         {
            // Initialize the map
            var map = new Microsoft.Maps.Map(document.getElementById("mapDiv"),
                         {credentials:"Your Bing Maps Key"});

            // Retrieve the location of the map center
            var center = map.getCenter();

            // Add a pin to the center of the map
            var pin = new Microsoft.Maps.Pushpin(center, {text: '1'});
            map.entities.push(pin);
         }

      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>
   </body>
</html>
```

To add a pushpin to a custom latitude and longitude coordinate, pass the Location object to the Pushpin constructor, then set the view based on the location as shown below.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         var map = null;

         function GetMap()
         {
            // Initialize the map
            map = new Microsoft.Maps.Map(document.getElementById("myMap"),
                         {credentials:"Bing Maps Key"});

            // Define the pushpin location
            var loc = new Microsoft.Maps.Location(47.592, -122.332);

            // Add a pin to the map
            var pin = new Microsoft.Maps.Pushpin(loc);
            map.entities.push(pin);

            // Center the map on the location
            map.setView({center: loc, zoom: 10});


         }


      </script>
   </head>
   <body onload="GetMap();">
      <div id='myMap' style="position:relative; width:400px; height:400px;"></div>
   </body>
```

```
</html>
```

Adding a Shape

To add a polyline or a polygon, use the same method used to add a pushpin. First, create your shape then add it to the entities property of the map. The following code adds a purple polygon to the map.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         function GetMap()
         {
            // Initialize the map
            var map = new
Microsoft.Maps.Map(document.getElementById("mapDiv"),{credentials:"Your Bing Maps Key"});

            // Create a polygon
            var vertices = new Array(new Microsoft.Maps.Location(20,-20), new
Microsoft.Maps.Location(20,20), new Microsoft.Maps.Location(-20,20), new
Microsoft.Maps.Location(-20,-20), new Microsoft.Maps.Location(20,-20));
            var polygoncolor = new Microsoft.Maps.Color(100,100,0,100);
            var polygon = new Microsoft.Maps.Polygon(vertices,{fillColor: polygoncolor,
strokeColor: polygoncolor});

            // Add the polygon to the map
            map.entities.push(polygon);
         }
      </script>

   </head>

   <body onload="GetMap();">

      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>

   </body>

</html>
```

Adding Multiple Entities to the Map

If you want to add multiple entities to the map at one time, first create an EntityCollection then add this collection to the map. The code below adds a polygon with pushpins at its corners.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

   <head>

      <title></title>

      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">


      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>


      <script type="text/javascript">


         function GetMap()

         {

            // Initialize the map

            var map = new Microsoft.Maps.Map(document.getElementById("mapDiv"),
{credentials:"Your Bing Maps Key"});


            // Create the locations

            var location1 = new Microsoft.Maps.Location(20,-20);

            var location2 = new Microsoft.Maps.Location(20,20);

            var location3 = new Microsoft.Maps.Location(-20,20);

            var location4 = new Microsoft.Maps.Location(-20,-20);
            // Create a polygon
            var vertices = new Array(location1, location2, location3, location4,
location1);
            var polygoncolor = new Microsoft.Maps.Color(100,100,0,100);
            var polygon = new Microsoft.Maps.Polygon(vertices,{fillColor: polygoncolor,
strokeColor: polygoncolor});

            // Create the entity collection with the polygon and pushpins at each corner
            var polygonWithPins = new Microsoft.Maps.EntityCollection();
            polygonWithPins.push(polygon);
            polygonWithPins.push(new Microsoft.Maps.Pushpin(location1));
            polygonWithPins.push(new Microsoft.Maps.Pushpin(location2));
            polygonWithPins.push(new Microsoft.Maps.Pushpin(location3));
            polygonWithPins.push(new Microsoft.Maps.Pushpin(location4));

            // Add the shape to the map
            map.entities.push(polygonWithPins)

         }

      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>
   </body>
</html>
```

Changing Entities on the Map

Once entities have been added to the map, you can use the methods of the map entities collection to change and manipulate those entities. The code implements a button to change the color of a shape on the map.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
```

```
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

          var map = null;

          // Define colors
          var blue = new Microsoft.Maps.Color(100, 0, 0, 200);
          var green = new Microsoft.Maps.Color(100, 0, 100, 100);
          var purple = new Microsoft.Maps.Color(100, 100, 0, 100);


         function GetMap()
         {
            // Initialize the map
            map = new
Microsoft.Maps.Map(document.getElementById("mapDiv"),{credentials:"Your Bing Maps Key"});

            // Create the locations
            var location1 = new Microsoft.Maps.Location(20,-20);
            var location2 = new Microsoft.Maps.Location(20,20);
            var location3 = new Microsoft.Maps.Location(-20,20);
            var location4 = new Microsoft.Maps.Location(-20, -20);
            var location5 = new Microsoft.Maps.Location(40, 0);


            // Create some shapes
            var triangleVertices = new Array(location1, location2, location5, location1);
            var triangle = new Microsoft.Maps.Polygon(triangleVertices, { fillColor:
blue, strokeColor: blue });


            var squareVertices = new Array(location1, location2, location3, location4,
location1);
            var square = new Microsoft.Maps.Polygon(squareVertices,{fillColor: purple,
strokeColor:purple});


            // Add the shapes to the map
            map.entities.push(triangle);
            map.entities.push(square);
         }


         function ChangePolygonColor()
         {
            // Get the map square entity. We know square was the last entity added,
            //    so we can calculate the index.
             var mapSquare = map.entities.get(map.entities.getLength()-1);


            // Get the current color
            var currentColor = mapSquare.getFillColor();


            if((currentColor.toString()) == (purple.toString()))
            {
               // Change it to green
               mapSquare.setOptions({fillColor: green, strokeColor:green});
            }
            else
            {
               // Change it back to purple
               mapSquare.setOptions({fillColor:purple, strokeColor:purple});
            }
         }
      </script>
```

```
   </head>

   <body onload="GetMap();">

      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>

      <input id="btnChangeColor" type="button" value="Change Polygon Color"
onclick="ChangePolygonColor();"/>

   </body>

</html>
```

Customizing Your Pushpins

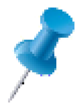
The Bing Maps AJAX Control, Version 7.0 provides flexible pushpin functionality. Use options provided in the PushpinOptions Object to customize your pushpins.

This topic describes how to customize your pushpin icon as well as how to create a pushpin info box.

Customizing Your Pushpin Icon

If you do not want to use the default pushpin icon, you can set the icon property of the PushpinOptions to the image you want to use instead.

This example uses the image below, named “BluePushpin.png”, as the pushpin icon.



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

   <head>

      <title></title>

      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">


      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>


      <script type="text/javascript">
```

```
         var map = null;

         function GetMap()
         {
            // Initialize the map
            map = new Microsoft.Maps.Map(document.getElementById("myMap"),
                         {credentials:"Bing Maps Key"});

            // Retrieve the location of the map center
            var center = map.getCenter();

            // Add a pin to the center of the map, using a custom icon
            var pin = new Microsoft.Maps.Pushpin(center, {icon: 'BluePushpin.png', width:
50, height: 50, draggable: true});

            map.entities.push(pin);
         }

      </script>
   </head>
   <body onload="GetMap();">
      <div id='myMap' style="position:relative; width:400px; height:400px;"></div>
   </body>
</html>
```

Creating a Pushpin Infobox

The Bing Maps AJAX Control, Version 7.0 has built-in pushpin info box functionality which you can customize to suit the needs of your application. To create an info box, use the Infobox and InfoboxOptions types.

The example below shows how to display an info box when a pushpin is clicked.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>


      <script type="text/javascript">

         var map = null;
         var pinInfobox = null;

         function GetMap()
         {
            // Initialize the map
            map = new Microsoft.Maps.Map(document.getElementById("myMap"),
{credentials:"Bing Maps Key"});

            // Retrieve the location of the map center
            var center = map.getCenter();

            // Add a pin to the center of the map
            var pin = new Microsoft.Maps.Pushpin(center, {text: '1'});

            // Create the infobox for the pushpin
            pinInfobox = new Microsoft.Maps.Infobox(pin.getLocation(),
                {title: 'My Pushpin',
                 description: 'This pushpin is located at (0,0).',
                 visible: false,
                 offset: new Microsoft.Maps.Point(0,15)});

            // Add handler for the pushpin click event.
            Microsoft.Maps.Events.addHandler(pin, 'click', displayInfobox);

            // Hide the infobox when the map is moved.
            Microsoft.Maps.Events.addHandler(map, 'viewchange', hideInfobox);



            // Add the pushpin and infobox to the map
            map.entities.push(pin);
            map.entities.push(pinInfobox);

         }

         function displayInfobox(e)
         {
            pinInfobox.setOptions({ visible:true });
         }

         function hideInfobox(e)
         {
            pinInfobox.setOptions({ visible: false });
         }
      </script>
   </head>
   <body onload="GetMap();">
      <div id='myMap' style="position:relative; width:500px; height:500px;"></div>
</html>
```

Displaying Location Search Results Using the REST Services

The Bing Maps AJAX Control, version 7.0 does not have built in functionality to return find results, but you can easily use the Bing Maps REST Services to geocode locations that you can then display on the map.

Initialize the Map

Before you add geocoding functionality, initialize the map using the following code.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         var map = null;

         function GetMap()
         {
            // Initialize the map
            map = new
Microsoft.Maps.Map(document.getElementById("mapDiv"),{credentials:"Your Bing Maps Key",
mapTypeId:"r"});

         }

      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>
   </body>
</html>
```

Add Controls

For this sample, add a text box and a Geocode button. In your script, create a ClickGeocode function that is called when the button is clicked.

```
<input id="txtQuery" type="text" value="Portland"/>

<input type="button" value="Geocode" onclick="ClickGeocode()"/>
```

Since the Bing Maps REST Services also require a Bing Maps Key, you need to first retrieve the key from the map object to ensure the session is valid. Use the `getCredentials` method of the Map Class to do this. The `getCredentials` method takes a function to call when the credentials are retrieved.

```
function ClickGeocode(credentials)

{

   map.getCredentials(MakeGeocodeRequest);

}
```

Make a Geocode REST Request

Next, make a geocode request to the Bing Maps REST Services using the value in the `txtQuery` input box and the credentials.

The Bing Maps REST Services can return an XML or JSON response object. For JavaScript code, JSON is more appropriate, so set `output=JSON`. This means that you need to also set a jsonp callback function name. In this sample the callback function is named `GeocodeCallback`. Finally, since you do not know if the text provided is a place name or an address, supply the `locationQuery` parameter and set it to the value of the `txtQuery` text box. Your REST geocode request looks like this:

```
var geocodeRequest = "http://dev.virtualearth.net/REST/v1/Locations/" +
document.getElementById('txtQuery').value + "?output=json&jsonp=GeocodeCallback&key=" +
credentials;
```

Now add script to make the REST request.

```
         function MakeGeocodeRequest(credentials)

         {

            var geocodeRequest = "http://dev.virtualearth.net/REST/v1/Locations/" +
document.getElementById('txtQuery').value + "?output=json&jsonp=GeocodeCallback&key=" +
credentials;


            CallRestService(geocodeRequest);

         }
         function CallRestService(request)
         {
            var script = document.createElement("script");
            script.setAttribute("type", "text/javascript");
            script.setAttribute("src", request);
            document.body.appendChild(script);
         }
         function GeocodeCallback(result)
         {
            // Do something with the result
         }
```

Display the Results

Finally, add code to the `GeocodeCallback` function to set the map view to the found location and add a pushpin at that location. The final code is shown below.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         var map = null;

         function GetMap()
         {
            // Initialize the map
            map = new
Microsoft.Maps.Map(document.getElementById("mapDiv"),{credentials:"Your Bing Maps Key",
mapTypeId:"r"});

         }

         function ClickGeocode(credentials)
         {
            map.getCredentials(MakeGeocodeRequest);
         }

         function MakeGeocodeRequest(credentials)
         {

            var geocodeRequest = "http://dev.virtualearth.net/REST/v1/Locations/" +
document.getElementById('txtQuery').value + "?output=json&jsonp=GeocodeCallback&key=" +
credentials;

            CallRestService(geocodeRequest);
         }

         function GeocodeCallback(result)
         {
            alert("Found location: " + result.resourceSets[0].resources[0].name);

            if (result &&
                   result.resourceSets &&
                   result.resourceSets.length > 0 &&
                   result.resourceSets[0].resources &&
                   result.resourceSets[0].resources.length > 0)
            {
               // Set the map view using the returned bounding box
               var bbox = result.resourceSets[0].resources[0].bbox;
               var viewBoundaries = Microsoft.Maps.LocationRect.fromLocations(new
Microsoft.Maps.Location(bbox[0], bbox[1]), new Microsoft.Maps.Location(bbox[2],
bbox[3]));
               map.setView({ bounds: viewBoundaries});

               // Add a pushpin at the found location
               var location = new
Microsoft.Maps.Location(result.resourceSets[0].resources[0].point.coordinates[0],
result.resourceSets[0].resources[0].point.coordinates[1]);
               var pushpin = new Microsoft.Maps.Pushpin(location);
               map.entities.push(pushpin);
            }
         }

         function CallRestService(request)
         {
            var script = document.createElement("script");
            script.setAttribute("type", "text/javascript");
            script.setAttribute("src", request);
            document.body.appendChild(script);
         }

      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>
      <input id="txtQuery" type="text" value="Portland"/>
      <input type="button" value="Geocode" onclick="ClickGeocode()"/>
   </body>
</html>
```

Getting Route Directions Using the REST Services

The Bing Maps AJAX Control, version 7.0 does not have built in route functionality, but you can easily use the Bing Maps REST Services to calculate a route and display it on the map.

Initialize the Map

Before you add route functionality, initialize the map using the following code.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

   <head>

      <title></title>

      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">


      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>


      <script type="text/javascript">


         var map = null;


         function GetMap()

         {

            // Initialize the map

            map = new
Microsoft.Maps.Map(document.getElementById("mapDiv"),{credentials:"Your Bing Maps Key",
mapTypeId:"r"});


         }


      </script>

   </head>

   <body onload="GetMap();">
```

```
      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>

   </body>

</html>
```

Construct the Route Request

Next, add input controls and construct the Bing Maps REST Services Route request.

In this example, a route is calculated between a specified start and end point. Add two text boxes and a button to initiate the route calculation.

```
<input id="txtStart" type="text" value="Seattle"/>

<input id="txtEnd" type="text" value="Portland"/>

<input type="button" value="Calculate Route" onclick="ClickRoute()"/>
```

Then, construct the REST route request using the input values.

```
var routeRequest = "http://dev.virtualearth.net/REST/v1/Routes?wp.0=" +
document.getElementById('txtStart').value + "&wp.1=" +
document.getElementById('txtEnd').value +
"&routePathOutput=Points&output=json&jsonp=RouteCallback&key=" + credentials;
```

Display the Results

Finally, add code to make the route request when the button is clicked, and add code to the `RouteCallback` function to set the map view and draw the route. The final code is shown below.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

   <head>

      <title></title>

      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">


      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>


      <script type="text/javascript">


         var map = null;
         function GetMap()
         {
            // Initialize the map
            map = new
Microsoft.Maps.Map(document.getElementById("mapDiv"),{credentials:"Your Bing Maps Key",
mapTypeId:"r"});



         }


         function ClickRoute(credentials)
         {

            map.getCredentials(MakeRouteRequest);
         }



         function MakeRouteRequest(credentials)
         {
            var routeRequest = "http://dev.virtualearth.net/REST/v1/Routes?wp.0=" +
document.getElementById('txtStart').value + "&wp.1=" +
document.getElementById('txtEnd').value +
"&routePathOutput=Points&output=json&jsonp=RouteCallback&key=" + credentials;


            CallRestService(routeRequest);


         }



          function RouteCallback(result) {



             if (result &&
                   result.resourceSets &&
                   result.resourceSets.length > 0 &&
                   result.resourceSets[0].resources &&
                   result.resourceSets[0].resources.length > 0) {

                     // Set the map view
                     var bbox = result.resourceSets[0].resources[0].bbox;
                     var viewBoundaries = Microsoft.Maps.LocationRect.fromLocations(new
Microsoft.Maps.Location(bbox[0], bbox[1]), new Microsoft.Maps.Location(bbox[2],
bbox[3]));
                     map.setView({ bounds: viewBoundaries});



                     // Draw the route
                     var routeline = result.resourceSets[0].resources[0].routePath.line;
                     var routepoints = new Array();

                     for (var i = 0; i < routeline.coordinates.length; i++) {

                         routepoints[i]=new
Microsoft.Maps.Location(routeline.coordinates[i][0], routeline.coordinates[i][1]);
                     }



                     // Draw the route on the map
                     var routeshape = new Microsoft.Maps.Polyline(routepoints,
{strokeColor:new Microsoft.Maps.Color(200,0,0,200)});
                     map.entities.push(routeshape);


                 }
         }



         function CallRestService(request)
         {
            var script = document.createElement("script");

            script.setAttribute("type", "text/javascript");

            script.setAttribute("src", request);

            document.body.appendChild(script);

         }


      </script>

   </head>

   <body onload="GetMap();">

      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>

      <input id="txtStart" type="text" value="Seattle"/>

      <input id="txtEnd" type="text" value="Portland"/>

      <input type="button" value="Calculate Route" onclick="ClickRoute()"/>

   </body>

</html>
```

Working with Tile Layers

This topic describes how to add a custom tile layer to the map.

Adding a Tile Layer

A tile layer is a valid map entity, so after you construct your layer, you can add it to the map using the `push` method of the map entities collection. The code below adds a custom tile layer to the map.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

   <head>

      <title></title>

      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">


      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>
```

```
      <script type="text/javascript">

         function GetMap()
         {
            // Initialize the map
            var map = new
Microsoft.Maps.Map(document.getElementById("mapDiv"),{credentials:"Your Bing Maps Key",
center:new Microsoft.Maps.Location(48.03,-122.4), zoom:12, mapTypeId:"r"});

            try
            {
               // Create the tile layer source
               var tileSource = new Microsoft.Maps.TileSource({uriConstructor:
'http://www.microsoft.com/maps/isdk/ajax/layers/lidar/{quadkey}.png'});

               // Construct the layer using the tile source
               var tilelayer= new Microsoft.Maps.TileLayer({ mercator: tileSource,
opacity: .7 });

               // Push the tile layer to the map
               map.entities.push(tilelayer);

            }
            catch(err)
            {
               alert( 'Error Message:' + err.message);
            }
         }

      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>
   </body>
```

```
</html>
```

Using Events in the AJAX Control

The Bing Maps AJAX Control 7.0 provides many events to allow your application to respond to user actions. The EntityCollection, Map, Pushpin, Polyline, and Polygon classes all have event members. The code examples in this topic show how to use the Map `click` event and the EntityCollection `entityadded` event.

Example

The example below shows how to use the Map `click` event to display the coordinate values of the clicked point in a text box.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         var map = null;

         function GetMap()
         {
            // Initialize the map
            map = new Microsoft.Maps.Map(document.getElementById("myMap"),
                          {credentials:"Bing Maps Key"});

            //Add handler for the map click event.
            Microsoft.Maps.Events.addHandler(map, 'click', displayLatLong);
         }

          function displayLatLong(e) {
              if (e.targetType == "map") {
                  var point = new Microsoft.Maps.Point(e.getX(), e.getY());
                  var loc = e.target.tryPixelToLocation(point);
                  document.getElementById("textBox").value= loc.latitude + ", " +
loc.longitude;

              }
          }

      </script>
   </head>
   <body onload="GetMap();">
      <div id='myMap' style="position:relative; width:400px; height:400px;"></div><br>
      <b>Click the map to display the coordinate values at that point.</b><br>
      Latitude, Longitude:
      <input id='textBox' type="text" style="width:250px;"/>
   </body>
</html>
```

You can expand the example above and add a pushpin wherever the user clicks. The code below also “greys out” the other pushpins in the entities collection when a new one is added.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">
```

```
var map = null;
var noPins = true;

function GetMap()
{
   // Initialize the map
   map = new Microsoft.Maps.Map(document.getElementById("myMap"),
                 {credentials:"Bing Maps Key"});

   // Add a handler for the map click event.
   Microsoft.Maps.Events.addHandler(map, 'click', addPin);

   // Add a handler to function that will grey out
   //    other pins in the collection when a new one is added
   Microsoft.Maps.Events.addHandler(map.entities, 'entityadded', shadePins);


}

 function addPin(e) {
     if (e.targetType == "map") {
         var point = new Microsoft.Maps.Point(e.getX(), e.getY());
         var loc = e.target.tryPixelToLocation(point);
         var pin = new Microsoft.Maps.Pushpin(loc);

         map.entities.push(pin);
     }
 }


 function shadePins(e) {

     if (noPins) {
```

```
                // If there aren't yet any pins on the map, do not grey the pin out.
                noPins = false;

             }
             else
             {
                 var pin = null;

                 // Loop through the collection of pushpins on the map and grey out
                 //    all but the last one added (which is at the end of the array).
                 var i = 0;
                 for (i = 0; i < e.collection.getLength() - 1; i++)
                 {
                     pin = e.collection.get(i);
                     pin.setOptions({ icon: "GreyPin.png" });
                 }
             }
         }

      </script>
   </head>
   <body onload="GetMap();">
      <div id='myMap' style="position:relative; width:400px; height:400px;"></div><br>
      <b>Click the map to add a pushpin at that point.</b><br>
   </body>
</html>
```

Returning a Localized Map

The Bing Maps AJAX Control 7.0 provides the ability to return a localized map.

Setting the Culture

By default the map labels and the navigation control text are provided in the culture English-United States (en-US). However, the map control culture can be changed by adding the **mkt** parameter to the map control reference, as in the following example, which sets the culture to French-France (fr-FR).

```
<script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0&mkt=fr-
fr"></script>
```

Supported Cultures

The following table lists supported cultures for the map control. The Culture column lists the valid values for the **mkt** parameter.

Language - Country/Region	**Culture**
Dutch - Belgium	nl-BE
English - Canada	en-CA
English - India	en-IN
English - United Kingdom	en-GB
English - United States	en-US
French - Canada	fr-CA
French - France	fr-FR
German - Germany	de-DE
Italian - Italy	it-IT
Japanese - Japan	ja-JP
Spanish - Mexico	es-MX
Spanish - Spain	es-ES
Spanish – United States	es-US

Remarks

- Error messages are always displayed in English - United States.

Bing Maps AJAX 7.0 Control API Reference

This section contains reference documentation for the Bing Maps AJAX Control 7.0.

In This Section

The Bing Maps AJAX Control 7.0 contains the following classes and enumerations.

Data Structures

- AltitudeReference Enumeration
- Location Class
- LocationRect Class
- Point Class

Mapping

- Events Object
- KeyEventArgs Object
- LabelOverlay Enumeration
- Map Class
- MapOptions Object
- MapTypeId Enumeration
- MouseEventArgs Object
- PixelReference Enumeration
- ViewOptions Object

Entities

- Color Class
- EntityCollection Class
- EntityCollectionOptions Object
- Events Object
- Infobox Class
- InfoboxOptions Object
- MouseEventArgs Object
- Polyline Class
- PolylineOptions Object
- Polygon Class
- PolygonOptions Object
- Pushpin Class

- PushpinOptions Object
- TileLayer Class
- TileLayerOptions Object
- TileSource Class
- TileSourceOptions Object

AltitudeReference Enumeration

Defines the reference point from which the altitude is measured.

Constants

Name	Description
ground	The altitude is measured from the ground level.
ellipsoid	The altitude is measured from the WGS 84 ellipsoid of the Earth.

Methods

Name	Definition	Return Value	Description
isValid	`isValid(reference:`AltitudeReference`)`	*boolean*	Determines if the specified reference is a supported AltitudeReference.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

   <head>

      <title></title>

      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>


      <script type="text/javascript">
      function GetMap()
      {
         // Create two locations with different altitude references.
         var groundLoc = new Microsoft.Maps.Location(47, -122, 100,
Microsoft.Maps.AltitudeReference.ground);
         var ellipsoidLoc = new Microsoft.Maps.Location(47, -122, 100,
Microsoft.Maps.AltitudeReference.ellipsoid);



         // Set the map view
         var mapOptions = {credentials: "Bing Maps Key",
                           center: groundLoc,
                           mapTypeId: Microsoft.Maps.MapTypeId.birdseye,
                           zoom:16};


         var map = new Microsoft.Maps.Map(document.getElementById("mapDiv"), mapOptions);


         // Add two pushpins to demonstrate the difference when using different altitude
references
         map.entities.push(new Microsoft.Maps.Pushpin(groundLoc, {text: "G"}));
         map.entities.push(new Microsoft.Maps.Pushpin(ellipsoidLoc, {text: "E"}));


      }
      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative; width:600px; height:600px;"></div>
   </body>
</html>
```

Color Class

Represents a color.

Constructor

Name	Definition	Description
Color	`Color(a:`*number*`, r:`*number*`, g:`*number*`, b:`*number*`)`	Initializes a new instance of the Color class. The `a` parameter represents opacity. The range of valid values for all parameters is 0 to 255.

Properties

Name	Type	Description
a	*number*	The opacity of the color. The range of valid values is 0 to 255.
r	*number*	The red value of the color. The range of valid values is 0 to 255.
g	*number*	The green value of the color. The range of valid values is 0 to 255.
b	*number*	The blue value of the color. The range of valid values is 0 to 255.

Static Methods

Name	Definition	Return Value	Description
clone	`clone(color:`Color`)`	Color	Creates a copy of the Color object.
fromHex	`fromHex(hex:`*string*`)`	Color	Converts the specified `hex` string to a Color.

Methods

Name	Definition	Return Value	Description
clone	`clone()`	Color	Returns a copy of the Color object.
getOpacity	`getOpacity()`	*number*	Returns the opacity of the Color as a value between 0 (a=0) and 1 (a=255).
toHex	`toHex()`	*string*	Converts the Color into a 6-digit hex string. Opacity is ignored. For example, a Color with values (255,0,0,0) is returned as hex string *#000000*.
toString	`toString()`	*string*	Converts the Color object to a string.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

   <head>

      <title></title>

      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">


      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>


      <script type="text/javascript">


          var map = null;
```

```
         function GetMap()
         {
            // Initialize the map
            map = new
Microsoft.Maps.Map(document.getElementById("mapDiv"),{credentials:"Your Bing Maps Key"});

            // Create the locations
            var location1 = new Microsoft.Maps.Location(-20,-20);
            var location2 = new Microsoft.Maps.Location(20,-20);
            var location3 = new Microsoft.Maps.Location(20,20);
            var location4 = new Microsoft.Maps.Location(60, 20);
            var location5 = new Microsoft.Maps.Location(60, 60);



            // Create a shape
            var lineVertices = new Array(location1, location2, location3, location4,
location5);
            var line = new Microsoft.Maps.Polyline(lineVertices,{strokeColor:new
Microsoft.Maps.Color(100, 100, 0, 100)});

            // Add the shape to the map
            map.entities.push(line);

         }

         function SetPolygonColor()
         {
            // Get the polyline entity.
            var mapLine = map.entities.get(0);

            // Set the option values
            var opacity = document.getElementById('txtA').value;
            var rValue = document.getElementById('txtR').value;
            var gValue = document.getElementById('txtG').value;
            var bValue = document.getElementById('txtB').value;
            var lineWidth = document.getElementById('txtWidth').value;


            // Verify input values and set the opacity, color,
            //    and width of the line.
            if (((opacity < 0) || (opacity > 255)) ||
                ((rValue < 0) || (rValue > 255)) ||
                ((gValue < 0) || (gValue > 255)) ||
                ((bValue < 0) || (bValue > 255)) )
            {
               alert("Opacity and all color values must be between 0 and 255.");
            }
            else
            {
               mapLine.setOptions({strokeColor:new Microsoft.Maps.Color(opacity, rValue,
gValue, bValue), strokeThickness:lineWidth});
            }

         }


      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>
      Line Opacity: <input id="txtA" type="text" value="100"/><br/>
      Red Color Value: <input id="txtR" type="text" value="100"/><br/>
      Green Color Value: <input id="txtG" type="text" value="100"/><br/>
      Blue Color Value: <input id="txtB" type="text" value="100"/><br/>
      Line Width: <input id="txtWidth" type="text" value="5"/><br/>
      <input id="btnChangeColor" type="button" value="Set Polygon Color"
onclick="SetPolygonColor();"/>
   </body>
```

```
</html>
```

EntityCollection Class

Contains a collection of entities. An Entity can be any one of the following types: Polygon, Polyline, Pushpin, TileLayer, or EntityCollection.

Constructor

Name	Definition	Description
EntityCollection	`EntityCollection(options?:`EntityCollectionOptions`)`	Initializes a new instance of the EntityCollection class.

Methods

Name	Definition	Return Value	Description
clear	`clear()`	None	Removes all entities from the collection.
get	`get(index:`*number*`)`	*Entity**	Returns the entity at the specified index in the collection.
getLength	`getLength()`	*number*	Returns the number of entities in the collection.
getVisible	`getVisible()`	*boolean*	Returns whether the entity collection is visible on the map.
getZIndex	`getZIndex()`	*number*	Gets the z-index of the entity collection with

Name	Definition	Return Value	Description
			respect to other items on the map.
indexOf	**`indexOf`**`(entity:`*Entity**`)`	*number*	Returns the index of the specified entity in the collection. If the entity is not found in the collection, -1 is returned.
insert	**`insert`**`(entity:`*Entity**`, index:`*number*`)`	None	Inserts the specified entity into the collection at the given index.
pop	**`pop`**`()`	*Entity**	Removes the last entity from the collection and returns it.
push	**`push`**`(entity:`*Entity**`)`	None	Adds the specified entity to the last position in the collection.
remove	**`remove`**`(entity:`*Entity**`)`	*Entity**	Removes the specified entity from the collection and returns it.
removeAt	**`removeAt`**`(index:`*number*`)`	*Entity**	Removes the entity at the specified index from the collection and returns it.
setOptions	**`setOptions`**`(options:`EntityCollectionOptions`)`	None	Sets the options for the entity collection.
toString	**`toString`**`()`	*string*	Converts the EntityCollection object to a string.

* An Entity can be any one of the following types: Infobox, Polygon, Polyline, Pushpin, TileLayer, and EntityCollection.

Events

Name	Arguments	Description
entityadded	*object*: {collection: EntityCollection, entity:*Entity**}	Occurs when one of the following happens: • An `entity` is added to the `collection`. • One of the entities of the collection (such as another entity collection) fires the **entityadded** event. For example, if collection #1 contains an entity, which is another collection #2, then when an entity is added to collection #2, two **entityadded** events are fired.
entitychanged	*object*: {collection: EntityCollection, entity:*Entity**}	Occurs when one of the following happens: • The `collection` changes. • An `entity` of the `collection` changes. • One of the entities of the collection (such as another entity collection) fires the **entitychanged** event. For example, if collection #1 contains an entity, which is another collection #2, then when an entity of collection #2 changes, two **entitychanged** events are fired.
entityremoved	*object*: {collection: EntityCollection, entity:*Entity**}	Occurs when one of the following happens: • An `entity` of the `collection` is removed. • One of the entities of the

Name	Arguments	Description
		collection (such as another entity collection) fires the **entityremoved** event. For example, if collection #1 contains an entity, which is another collection #2, then when an entity of collection #2 is removed, two **entityremoved** events are fired.

* An Entity can be any one of the following types: Infobox, Polygon, Polyline, Pushpin, TileLayer, and EntityCollection.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

   <head>

      <title></title>

      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">


      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>


      <script type="text/javascript">


         var map = null;

         var pinTotal = 0;


         function GetMap()

         {

            // Initialize the map

            map = new Microsoft.Maps.Map(document.getElementById("mapDiv"),
{credentials:"Bing Maps Key"});
```

```
   // Add handler for the map click event - add a pin to the click location.
   Microsoft.Maps.Events.addHandler(map, 'click', addPin);



}

function addPin(e) {

     if (e.targetType == "map") {

         // Return the location where the map was clicked and create the pin.
         var point = new Microsoft.Maps.Point(e.getX(), e.getY());
         var loc = e.target.tryPixelToLocation(point);
         var pin = new Microsoft.Maps.Pushpin(loc);

         // Attach a handler to the pin so that it is removed when it is clicked
         Microsoft.Maps.Events.addHandler(pin, 'click', removePin);



         // Add the pushpin
         map.entities.push(pin);



     }
}

function removePin(e)
{
     var indexOfPinToRemove = map.entities.indexOf(e.target);

     map.entities.removeAt(indexOfPinToRemove);

}
```

```
      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>
      <b>Click the map to add a pin, or click a pin to remove it.<b/>
   </body>
</html>
```

EntityCollectionOptions Object

Contains options for an entity collection.

Properties

Name	Type	Description
visible	*boolean*	A Boolean indicating whether the entity collection is visible on the map.
zIndex	*number*	The z-index of the entity collection with respect to other items on the map.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">
         var map = null;

         function GetMap()
         {
            // Initialize the map
            map = new
Microsoft.Maps.Map(document.getElementById("mapDiv"),{credentials:"Bing Maps Key"});

            // Add handler for the map click event - add a pin to the click location.
            Microsoft.Maps.Events.addHandler(map, 'click', addPin);



         }

         function addPin(e) {
              if (e.targetType == "map") {

                  // Return the location where the map was clicked and create the pin.
                  var point = new Microsoft.Maps.Point(e.getX(), e.getY());
                  var loc = e.target.tryPixelToLocation(point);
                  var pin = new Microsoft.Maps.Pushpin(loc);

                  // Add the pushpin
                  map.entities.push(pin);



              }
         }

         function hideAllPins(){
              // Hide all the entities on the map
              map.entities.setOptions({visible:false});
         }



         function showAllPins(){
              // Show all the entities on the map
              map.entities.setOptions({visible:true});
         }



      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>
      <input type="button" value="Hide all pins" onclick="hideAllPins();"/>
      <input type="button" value="Show all pins" onclick="showAllPins();"/>
   </body>
</html>
```

Events Object

Provides event handling functionality for map and entity events.

The Events object does not need to be initialized. Call the Events methods directly.

Methods

Name	Definition	Return Value	Description
addHandler	**`addHandler`**`(target:`*object*, `eventName:`*string*, `handler:`*function*`)`	*object*	Attaches the handler for the event that is thrown by the target. Use the return object to remove the handler using the **removeHandler**

Name	Definition	Return Value	Description
			method. `Microsoft.Maps.Events.addHandler(map, 'viewchangedend', function(e){ //Handle the event });`
addThrottledHandler	**`addThrottledHandler`**`(target:`*object*`, eventName:`*string*`, handler:`*function*`, throttleInterval:`*number*`)`	*object*	Attaches the handler for the event that is thrown by the target, where the minimum interval between events (in milliseconds) is specified in the `throttleInterval` parameter. The last occurrence of the event is called after the specified `throttleInterval`.
hasHandler	**`hasHandler`**`(target:`*object*`, eventName:`*string*`)`	*boolean*	Checks if the target has any attached event handler.
invoke	**`invoke`**`(target:`*object*`, eventName:`*string*`, args:`*object*`)`	None	Invokes an event on the target. This causes all handlers for the specified `eventName` to be called.
removeHandler	**`removeHandler`**`(handlerId:`*object*`)`	None	Detaches the specified handler from the event. The `handlerId` is returned by the **addHandler** and **addThrottledHandler** methods.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>
```

```
      <script type="text/javascript">

         var map = null;
         var infobox = null;

         function GetMap()
         {
            // Initialize the map
            map = new Microsoft.Maps.Map(document.getElementById("myMap"),
                         {credentials:"Bing Maps Key"});

            // Retrieve the location of the map center
            var center = map.getCenter();

            // Create a pin at the center of the map and its corresponding infobox
            var pin = new Microsoft.Maps.Pushpin(center);
            infobox = new Microsoft.Maps.Infobox(center, {title: 'Pushpin infobox',
visible:false, offset:new Microsoft.Maps.Point(0,35)});

            // Add event handlers for hovering over the pushpin
            Microsoft.Maps.Events.addHandler(pin, 'mouseover', showInfobox);
            Microsoft.Maps.Events.addHandler(pin, 'mouseout', hideInfobox);

            // Add the pushpin and hidden infobox to the map
            map.entities.push(pin);
            map.entities.push(infobox);

         }

         function showInfobox()
         {
            infobox.setOptions({visible:true});

         }
         function hideInfobox()
         {
            infobox.setOptions({visible:false});


         }



      </script>
   </head>
   <body onload="GetMap();">
      <div id='myMap' style="position:relative; width:500px; height:500px;"></div>
   </body>
</html>
```

Infobox Class

Represents an info box on the map. You can use this class to create pop-up balloons for pushpins.

Constructor

Name	Definition	Description
Infobox	**`Infobox`**`(location:`Location`, options?:`InfoboxOptions`)`	Initializes a new instance of the Infobox class.

Methods

Name	Definition	Return Value	Description
getActions	**`getActions`**`()`	*Object*	Returns a list of actions, where each item is a name-value pair indicating an action link name and the event name for the

Name	Definition	Return Value	Description
			action that corresponds to that action link.
getAnchor	`getAnchor()`	Point	Returns the point on the infobox which is anchored to the map. An anchor of (0,0) is the top left corner of the infobox.
getDescription	`getDescription()`	*string*	Returns the string that is printed inside the infobox.
getHeight	`getHeight()`	*number*	Returns the height of the infobox.
getHtmlContent	`getHtmlContent()`	*string*	Returns the infobox as HTML.
getId	`getId()`	*string*	Returns the ID of the infobox.
getLocation	`getLocation()`	Location	Returns the location on the map where the infobox's anchor is attached.
getOffset	`getOffset()`	*number*	Returns the amount the infobox pointer is shifted from the location of the infobox, or if **showPointer** is false, then it is the amount the infobox bottom left edge is shifted from the location of the infobox. The default value is (0,0), which means there is no offset.
getOptions	`getOptions()`	InfoboxOptions	Returns the infobox options.
getShowCloseButton	`getShowCloseButton()`	*boolean*	Returns a boolean indicating whether the infobox close button is shown.
getShowPointer	`getShowPointer()`	*boolean*	Returns a boolean indicating whether the infobox is drawn with a pointer.
getTitle	`getTitle()`	*string*	Returns a string that is the title of the infobox.
getTitleClickHan	`getTitleClickHandler()`	*string*	Returns the name of the function

Name	Definition	Return Value	Description
dler			to call when the title of the infobox is clicked.
getVisible	`getVisible()`	*boolean*	Returns whether the infobox is visible. A value of **false** indicates that the infobox is hidden, although it is still an entity on the map.
getWidth	`getWidth()`	*number*	Returns the width of the infobox.
getZIndex	`getZIndex()`	*number*	Returns the z-index of the infobox with respect to other items on the map.

setHtmlContent	`setHtmlContent(content:`*string*`)`	None	Sets the HTML content of the infobox. You can use this method to change the look of the infobox.
			`var infoboxOptions = {width :200, height :100, showCloseButton: true, zIndex: 0, offset:new Microsoft.Maps.Point(10,0), showPointer: true};`
			`var defInfobox = new Microsoft.Maps.Infobox(map.getCenter(), infoboxOptions );`
			`map.entities.push(defInfobox);`
			`defInfobox.setHtmlContent('<div id="infoboxText" style="background-color:White; border-style:solid;border-width:medium; border-color:DarkOrange; min-height:100px; position:absolute;top:0px; left:23px; width:240px;"><b id="infoboxTitle" style="position:absolute; top:10px; left:10px; width:220px;">myTitle</b><a`

Name	Definition	Return Value	Description
			`id="infoboxDescription" style="position:absolute; top:30px; left:10px; width:220px;">lkjsl lkjdkl lkajdlkj klasdjfkl</a></div>');`
setLocation	`setLocation(location:`Location`)`	None	Sets the location on the map where the anchor of the infobox is attached.
setOptions	`setOptions(options:`InfoboxOptions`)`	None	Sets options for the infobox.
toString	`toString()`	*string*	Converts the Infobox object to a string.

Events

Name	Arguments	Description
click	*eventArgs:*MouseEventArgs	Occurs when the mouse is used to click the infobox.
entitychanged	*object*: {entity:*Entity*}	Occurs when the location of the infobox or any of the infobox options change.
mouseenter	*eventArgs:*MouseEventArgs	Occurs when the mouse cursor enters the area covered by the infobox.
mouseleave	*eventArgs:*MouseEventArgs	Occurs when the mouse cursor leaves the area covered by the infobox.

Remarks

- The Bing Maps AJAX Control default info box is designed for desktop browsers and may not function properly on all mobile browsers.
- For the best performance, it is recommended that you have only one info box on the map at a time.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>



      <script type="text/javascript">

         var map = null;
         var pinInfobox = null;

         function GetMap()
         {
            // Initialize the map
            map = new Microsoft.Maps.Map(document.getElementById("myMap"),
{credentials:"Bing Maps Key"});

            // Retrieve the location of the map center
            var center = map.getCenter();

            // Add a pin to the center of the map
            var pin = new Microsoft.Maps.Pushpin(center, {text: '1'});

            // Create the info box for the pushpin
            pinInfobox = new Microsoft.Maps.Infobox(new Microsoft.Maps.Location(0,0),
{title: 'My Pushpin', visible: true});
```

```
            // Add a handler for the pushpin click event.
            Microsoft.Maps.Events.addHandler(pin, 'click', displayInfobox);

            // Hide the info box when the map is moved.
            Microsoft.Maps.Events.addHandler(map, 'viewchange', hideInfobox);



            // Add the pushpin and info box to the map
            map.entities.push(pin);
            map.entities.push(pinInfobox);

         }

         function displayInfobox(e)
         {
            pinInfobox.setOptions({ visible:true });
         }



         function hideInfobox(e)
         {
            pinInfobox.setOptions({ visible: false });
         }



      </script>
   </head>
   <body onload="GetMap();">
      <div id='myMap' style="position:relative; width:500px; height:500px;"></div>
</html>
```

InfoboxOptions Object

Represents the options for an infobox.

Properties

Name	Type	Description
actions	*Object*	A list of the info box actions, where each item is a *label* (the action link text) and *eventHandler* (name of the function handling a click of the action link). `var infoboxOptions = {title:'My Infobox', description:'Testing actions', actions:[{label: 'test1', eventHandler: testEvent1}, {label: 'test2', eventHandler: testEvent2},{label: 'test3', eventHandler: testEvent3}]  };`
description	*string*	The string displayed inside the info box.
height	*number*	The height of the info box. The default value is 126.
htmlContent	*string*	The HTML that represents the info box. `var infoboxOptions = {width :200, height :100, showCloseButton: true, zIndex: 0, offset:new Microsoft.Maps.Point(10,0), showPointer: true, htmlContent:'<b>Custom HTML</b>'};`
id	*string*	The ID associated with the info box.
location	Location	The location on the map where the info box's anchor is attached.
offset	Point	The amount the info box pointer is shifted from the location of the info box, or if **showPointer** is false, then

Name	Type	Description
		it is the amount the info box bottom left edge is shifted from the location of the info box. The default value is (0,0), which means there is no offset.
showCloseButton	*boolean*	A boolean indicating whether to show the close dialog button on the info box. The default value is **true**. By default the close button is displayed as an **X** in the top right corner of the info box.
showPointer	*boolean*	A boolean indicating whether to display the info box with a pointer. The default value is **true**.
title	*string*	The title of the info box.
titleClickHandler	*string*	The name of the function to call when the title of the info box is clicked. If this property is set, the title of the info box is displayed as a link.
visible	*boolean*	A boolean indicating whether to show or hide the info box. The default value is **true**. A value of **false** indicates that the info box is hidden, although it is still an entity on the map.
width	*number*	The width of the info box. The default value is 256.
zIndex	*number*	The z-index of the info box with respect to other items on the map.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>


      <script type="text/javascript">

         var map = null;


         function GetMap()
         {
            // Initialize the map
            map = new Microsoft.Maps.Map(document.getElementById("myMap"),
{credentials:"Bing Maps Key"});

            // Retrieve the location of the map center
            var center = map.getCenter();



            // Create an info box
            var infoboxOptions = {width:300,
                                  height: 100,
                                  title: "Information Box Title",
                                  description: "This is the map.",
                                  showPointer: false,
                                  titleClickHandler: InfoboxHandler,
                                  offset: new Microsoft.Maps.Point(-100,0)};
            var myInfobox = new Microsoft.Maps.Infobox(center, infoboxOptions);



            // Add the info box to the map
            map.entities.push(myInfobox);
         }


         function InfoboxHandler()
         {
            alert("Infobox title was clicked!");
         }



      </script>
   </head>
   <body onload="GetMap();">
      <div id='myMap' style="position:relative; width:500px; height:500px;"></div>
</html>
```

See Also

Infobox Class

KeyEventArgs Object

Contains the arguments for keyboard events.

Properties

Name	Type	Description
altKey	*boolean*	A boolean indicating if the ALT key was pressed.
ctrlKey	*boolean*	A boolean indicating if the CTRL key was pressed.
eventName	*string*	The event that occurred.
handled	*boolean*	A boolean indicating whether the event is handled. If this property is set to **true**, the default map control behavior

Name	Type	Description
		for the event is cancelled.
keyCode	*string*	The ASCII character code that identifies the keyboard key that was pressed.
originalEvent	*object*	The original browser event.
shiftKey	*boolean*	A boolean indicating if the SHIFT key was pressed.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         var map = null;

         function GetMap()
         {
            // Initialize the map
            map = new Microsoft.Maps.Map(document.getElementById("myMap"),
                         {credentials:"Bing Maps Key"});

            //Add handler for the map click event.
            Microsoft.Maps.Events.addHandler(map, 'keypress', addPin);
         }

          function addPin(e) {

              if (e.keyCode == "112") {

                  // If the 'p' key is pressed, add a pushpin to the center of the
                  //   current map view.
                  var center = map.getCenter();
                  var pin = new Microsoft.Maps.Pushpin(center);
                  map.entities.push(pin);

              }
          }

      </script>
   </head>
   <body onload="GetMap();">
      <div id='myMap' style="position:relative; width:400px; height:400px;"></div><br>
      <b>Click the 'p' key to add a pushpin to the map.</b>
   </body>
</html>
```

LabelOverlay Enumeration

Defines how map labels are displayed.

Constants

Name	Description
hidden	Map labels are not shown on top of imagery.
visible	Map labels are shown on top of imagery.

Methods

Name	Definition	Return Value	Description
isValid	`isValid(labelOverlay:LabelOverlay)`	*boolean*	Determines whether the specified `labelOverlay` is a supported LabelOverlay.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         function GetMap()
         {
            // Initialize the map options.  In this case,
            //    turn the label overlay on a bird's eye map off.
            var mapOptions =
            {
               credentials:"Your Bing Maps Key",
               mapTypeId: Microsoft.Maps.MapTypeId.birdseye,
               center: new Microsoft.Maps.Location(37.794973,-122.393542),
               zoom: 17,
               labelOverlay: Microsoft.Maps.LabelOverlay.hidden
            }
            //Load the map

            var map = new Microsoft.Maps.Map(document.getElementById("mapDiv"),
mapOptions );


         }


      </script>

   </head>

   <body onload="GetMap();">

      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>

   </body>

</html>
```

Location Class (AJAX)

Contains the altitude and coordinate values of a location on the map.

Constructor

Name	Definition	Description
Location	`Location(latitude:`*number*, `longitude:`*number*, `altitude?:`*number*, `altitudeMode?:`AltitudeReference`)`	Initializes a new instance of the Location class. The `altitude` and `altitudeMode` parameters default to *undefined*.

Properties

Name	Type	Description
altitude	*number*	The altitude of the location.
altitudeMode	AltitudeReference	The reference from which the altitude is measured.
latitude	*number*	The latitude of the location.

Name	Type	Description
longitude	*number*	The longitude of the location.

Static Methods

Name	Definition	Return Value	Description
areEqual	`areEqual(location1:`Location`, location2:`Location`)`	*boolean*	Determines if the specified Location objects are equal.
normalizeLongitude	`normalizeLongitude(longitude:`*number*`)`	*number*	Normalizes the specified longitude so that it is between -180 and 180.

Methods

Name	Definition	Return Value	Description
clone	`clone()`	Location	Returns a copy of the Location object.
toString	`toString()`	*string*	Converts the Location object to a string.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

   <head>

      <title></title>

      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

      var map = null;

      function GetMap()
      {

         map = new Microsoft.Maps.Map(document.getElementById("mapDiv"), {credentials:
"Bing Maps Key"});

      }

      function SetLocation()
      {
         // Parse the input string
         var latLongArray = (document.getElementById("txtlatlong").value).split(",");

         // Retrieve the latitude and longitude values- normalize the longitude value
         var latVal = parseInt(latLongArray[0]);
         var longVal =
Microsoft.Maps.Location.normalizeLongitude(parseInt(latLongArray[1]));

         // Set the map center
         map.setView({center:new Microsoft.Maps.Location(latVal, longVal)});

      }
      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative; width:600px; height:600px;"></div>
```

```
        Latitude, Longitude: <input id="txtlatlong" type="text" value="40, -120"/>
        <input type="button" value="Set center location" onclick="SetLocation();"/>
    </body>
</html>
```

LocationRect Class

Represents a rectangle on the map.

Constructor

Name	Definition	Description
LocationRect	`LocationRect(center:`Location`, width:`*number*`, height:`*number*`)`	Initializes a new instance of the LocationRect class.

Properties

Name	Type	Description
center	Location	The location that defines the center of the rectangle.
height	*number*	The height, in degrees, of the rectangle.
width	*number*	The width, in degrees, of the rectangle.

Static Methods

Name	Definition	Return Value	Description
fromCorners	`fromCorners(northwest:`Location`, southeast:`Location`)`	LocationRect	Returns a LocationRect using the specified locations for the northwest and southeast corners.
fromEdge	`fromEdges(north:`*number*,	Location	Returns a LocationRect using the

Name	Definition	Return Value	Description
s	`west:`*number*, `south:`*number*, `east:`*number*, `altitude:`*number*, `altitudeReference:`Altitude Reference`)`	Rect	specified northern and southern latitudes and western and eastern longitudes for the rectangle boundaries.
fromLocation	**`fromLocation`**`(list of locations/array)`	Location Rect	Returns a LocationRect using a list of locations or an array of locations. To provide a list of locations: `Microsoft.Maps.LocationRect.fromLocations(location1, location2, location3);` To provide an array of locations: `var locations = [location1, location2, location3];` `Microsoft.Maps.LocationRect.fromLocations(locations);`
fromString	**`fromString`**`(string:`*string*`)`	Location Rect	Creates a LocationRect from a string with the following format: "north,west,south,east". North, west, south and east specify the coordinate number values.

Methods

Name	Definition	Return Value	Description
clone	**`clone`**`()`	LocationRect	Returns a copy of the LocationRect object.
contains	**`contains`**`(location:`Location`)`	*boolean*	Returns whether the specified Location is within the LocationRect.
getEast	**`getEast`**`()`	*number*	Returns the longitude that defines the

Name	Definition	Return Value	Description
			eastern edge of the LocationRect.
getNorth	`getNorth()`	*number*	Returns the latitude that defines the northern edge of the LocationRect.
getNorthwest	`getNorthwest()`	Location	Returns the Location that defines the northwest corner of the LocationRect.
getSouth	`getSouth()`	*number*	Returns the latitude that defines the southern edge of the LocationRect.
getSoutheast	`getSoutheast()`	Location	Returns the Location that defines the southeast corner of the LocationRect.
getWest	`getWest()`	*number*	Returns the latitude that defines the western edge of the LocationRect.
intersects	`intersects(rect:`LocationRect`)`	*boolean*	Returns whether the specified LocationRect intersects with this LocationRect.
toString	`toString()`	*string*	Converts the LocationRect object to a string.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

      var map = null;

      function GetMap()
      {

         map = new Microsoft.Maps.Map(document.getElementById("mapDiv"), {credentials:
"Bing Maps Key"});

         var viewRect = Microsoft.Maps.LocationRect.fromCorners(new
Microsoft.Maps.Location(40,-120), new Microsoft.Maps.Location(35,-115));

         map.setView({bounds: viewRect});

      }

      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative; width:500px; height:500px;"></div>
   </body>
</html>
```

Map Class

Represents a map.

Constructor

Name	Definition	Description
Map	`Map(containerElement:`*node*, `options?:`MapOptions or ViewOptions`)`	Initializes a new instance of the Map class.

Properties

Name	Type	Description
entities	EntityCollection	The map's entities. Use this property to add or remove entities from the map.

Methods

Name	Definition	Return Value	Description
blur	`blur()`	None	Removes focus from the map control so that it does not respond to keyboard events.
dispose	`dispose()`	None	Deletes the **Map** object and releases any associated resources.
focus	`focus()`	None	Applies focus to the map control so that it responds to keyboard events.
getBounds	`getBounds()`	LocationRect	Returns the location rectangle that defines the boundaries of the current map view.
getCenter	`getCenter()`	Location	Returns the location of the

Name	Definition	Return Value	Description
			center of the current map view.
getCopyrights	`getCopyrights()`	*string[]*	Gets the array of strings representing the attributions of the imagery currently displayed on the map.
getCredentials	`getCredentials(callback:`*function*`)`	None	Gets the session ID. This method calls the callback function with the session ID as the first parameter. `map.getCredentials(function(credentials) { if(credentials !== null) { /* Valid session Id. Use it to call REST service */ } });`
getHeading	`getHeading()`	*number*	Returns the heading of the current map view.
getHeight	`getHeight()`	*number*	Returns the height of the map control.
getImageryId	`getImageryId()`	*string*	Returns the string that represents the imagery currently displayed on the map.
getMapTypeId	`getMapTypeId()`	*string*	Returns a string that represents the current map type displayed on the map. Valid map types are listed in the MapTypeId Enumeration topic.
getMetersPerPixel	`getMetersPerPixel()`	*number*	Returns the current scale in meters per pixel of the center of the map.
getModeLayer	`getModeLayer()`	*Node*	Returns the map’s mode node.
getOptions	`getOptions()`	MapOptions	Returns the map options that

Name	Definition	Return Value	Description
			have been set. Note that if an option is not set, then the default value for that option is assumed and **getOptions** returns undefined for that option.
getPageX	`getPageX()`	*number*	Returns the x coordinate of the top left corner of the map control, relative to the page.
getPageY	`getPageY()`	*number*	Returns the y coordinate of the top left corner of the map control, relative to the page.
getRootElement	`getRootElement()`	*Node*	Returns the map's root node.
getTargetBounds	`getTargetBounds()`	LocationRect	Returns the location rectangle that defines the boundaries of the view to which the map is navigating.
getTargetCenter	`getTargetCenter()`	Location	Returns the center location of the view to which the map is navigating.
getTargetHeading	`getTargetHeading()`	*number*	Returns the heading of the view to which the map is navigating.
getTargetMetersPerPixel	`getTargetMetersPerPixel()`	*number*	Returns the scale in meters per pixel of the center of the view to which the map is navigating.
getTargetZoom	`getTargetZoom()`	*number*	Returns the zoom level of the view to which the map is navigating.
getUserLayer	`getUserLayer()`	*Node*	Returns the map's user node.
getViewportX	`getViewportX()`	*number*	Returns the x coordinate of the viewport origin (the center of the map), relative to the page.
getViewportY	`getViewportY()`	*number*	Returns the y coordinate of the viewport origin (the center of

Name	Definition	Return Value	Description
			the map), relative to the page.
getWidth	`getWidth()`	*number*	Returns the width of the map control.
getZoom	`getZoom()`	*number*	Returns the zoom level of the current map view.
getZoomRange	`getZoomRange()`	*object*:{min:*number*, max: *number*}	Returns the range of valid zoom levels for the current map view.
isMercator	`isMercator()`	*boolean*	Returns whether the map is in a regular Mercator nadir mode.
isRotationEnabled	`isRotationEnabled()`	*boolean*	Returns *true* if the current map type allows the heading to change; *false* if the display heading is fixed.
setMapType	`setMapType(mapTypeId:`*string*`)`	None	Sets the current map type. The specified `mapTypeId` must be a valid map type ID or a registered map type ID. Valid map type IDs are listed in the MapTypeId Enumeration topic.
setOptions	`setOptions({height:`*number*`, width: `*number*`})`	None	Sets the height and width of the map.
setView	`setView(options:`ViewOptions`)`	None	Sets the map view based on the specified options.
tryLocationToPixel	`tryLocationToPixel(location:`Location `|`Location`[], reference?:`PixelReference`)`	*null*, Point, or Point*[]*	Converts a specified Location to a Point on the map relative to the specified PixelReference. If `reference` is not specified, PixelReference.viewport is used. If the map is not able to convert the Location, *null* is returned. Alternatively, converts an array of Locations and returns an array of Points if all

Name	Definition	Return Value	Description
			locations were converted. If any of the conversions fail, *null* is returned.
tryPixelToLocation	`tryPixelToLocation(point:` Point | Point`[]`, `reference?:`PixelReference `)`	*null*, Location, or Location*[]*	Converts a specified Point to a Location on the map relative to the specified PixelReference. If `reference` is not specified, PixelReference.viewport is used. If the map is not able to convert the Point, *null* is returned. Alternatively, converts an array of Points and returns an array of Locations if all points were converted. If any of the conversions fail, *null* is returned.

Events

Name	Arguments	Description
click	*eventArgs:*MouseEventArgs	Occurs when the mouse is used to click the map.
copyrightchanged	None	Occurs when the copyright of the map changes.
dblclick	*eventArgs:*MouseEventArgs	Occurs when the mouse is used to double click the map.
imagerychanged	None	Occurs when the underlying imagery used by the map changes. This is different from the **maptypechanged** event, which occurs when the map type being used is changed.
keydown	*eventArgs:*KeyEventArgs	Occurs when a keyboard

Name	Arguments	Description
		key is pressed down.
keypress	*eventArgs:*KeyEventArgs	Occurs when a keyboard key is pressed.
keyup	*eventArgs:*KeyEventArgs	Occurs when a keyboard key that is pressed down is released.
maptypechanged	None	Occurs when the map type changes.
mousedown	*eventArgs:*MouseEventArgs	Occurs when the left mouse button is pressed when the mouse cursor is over the map.
mousemove	*eventArgs:*MouseEventArgs	Occurs when the mouse cursor moves over the map.
mouseout	*eventArgs:*MouseEventArgs	Occurs when the mouse cursor moves out of the area covered by the map.
mouseover	*eventArgs:*MouseEventArgs	Occurs when the mouse is over the map.
mouseup	*eventArgs:*MouseEventArgs	Occurs when the left mouse button is lifted up when the mouse cursor is over the map.
mousewheel	*eventArgs:*MouseEventArgs	Occurs when the mouse wheel is used when the mouse cursor is over the map.
rightclick	*eventArgs:*MouseEventArgs	Occurs when the right mouse button is used to click the map.
targetviewchanged	None	Occurs when the view towards which the map is navigating changes.
tiledownloadcomplete	None	Occurs when all the map tiles of a map view have loaded.

Name	Arguments	Description
viewchange	None	Occurs for every frame of a map view change.
viewchangeend	None	Occurs when the map view is done changing. This event occurs when a view is the same for one frame of a map view change. For example, if the mouse is used to drag the map to change the view, but pauses during the drag (without releasing the mouse button), **viewchangeend** occurs twice. You can use the addThrottledHandler method to customize the number of events that occur.
viewchangestart	None	Occurs when the map view starts changing.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         var map = null;
```

```
            function GetMap()
            {
               // Initialize the map
               map = new Microsoft.Maps.Map(document.getElementById("myMap"),
                            {credentials:"Bing Maps Key"});

            }

         </script>
      </head>
      <body onload="GetMap();">
         <div id='myMap' style="position:relative; width:400px; height:400px;"></div>
      </body>
   </html>
```

MapOptions Object

Represents options to customize the map that is displayed.

Properties

Name	Type	Description
credentials	*string*	The Bing Maps Key used to authenticate the application. This property is required and can only be set when using the Map constructor.
disableKeyboardInput	*boolean*	A boolean value indicating whether to disable the map's response to keyboard input. The default value is **false**. This property can only be set when using the Map constructor.

Name	Type	Description
disableMouseInput	*boolean*	A boolean value indicating whether to disable the map's response to mouse input. The default value is **false**. This property can only be set when using the Map constructor.
disableTouchInput	*boolean*	A boolean value indicating whether to disable the map's response to touch input. The default value is **false**. This property can only be set when using the Map constructor.
disableUserInput	*boolean*	A boolean value indicating whether to disable the map's response to any user input. The default value is **false**. This property can only be set when using the Map constructor.
enableClickableLogo	*boolean*	A boolean value indicating whether the Bing^{TM} logo on the map is clickable. The default value is **true**. This property can only be set when using the Map constructor.
enableSearchLogo	*boolean*	A boolean value indicating whether to enable the Bing^{TM} hovering search logo on the map. The default value is **true**. This property can only be set when using the Map constructor.
height	*number*	The height of the map. The default value is **null**. If no height is specified, the height of the div is used. If **height** is

Name	Type	Description
		specified, then **width** must be specified as well.
showCopyright	*boolean*	A boolean value indicating whether or not to show the map copyright. The default value is **true**. This property can only be set when using the Map constructor.
showDashboard	*boolean*	A boolean value indicating whether to show the map navigation control. The default value is **true**. This property can only be set when using the Map constructor.
showMapTypeSelector	*boolean*	A boolean value indicating whether to show the map type selector in the map navigation control. The default value is **true**. This property can only be set when using the Map constructor.
showScalebar	*boolean*	A boolean value indicating whether to show the scale bar. The default value is **true**. This property can only be set when using the Map constructor.
width	*number*	The width of the map. The default value is **null**. If no width is specified, the width of the div is used. If **width** is specified, then **height** must be specified as well.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         function GetMap()
         {

            // Set the map and view options, setting the map style to Road and
            //    removing the user's ability to change the map style
            var mapOptions = {credentials:"Bing Maps Key",
                              height: 400,
                              width: 400,
                              mapTypeId: Microsoft.Maps.MapTypeId.road,
                              showMapTypeSelector: false};

            // Initialize the map
            var map = new Microsoft.Maps.Map(document.getElementById("mapDiv"),
mapOptions);

         }

      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative;"></div>
   </body>
</html>
```

MapTypeId Enumeration

Contains identifiers for the imagery displayed on the map.

Constants

Name	Description
aerial	The aerial map style is being used.
auto	The map is set to choose the best imagery for the current view.
birdseye	The bird’s eye map type is being used.
collinsBart	Collin’s Bart (mkt=en-gb) map type is being used.
mercator	The Mercator style is being used.
ordnanceSurvey	Ordinance Survey (mkt=en-gb) map type is being used.
road	The road map style is being used.

Example

This code sample sets the map imagery to Collin’s Bart (**collinsBart**). Since this imagery is only supported for the en-gb culture, the *mkt* parameter of the control is set to this culture.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0&mkt=en-
gb"></script>

      <script type="text/javascript">
      function GetMap()
      {

         var map = new Microsoft.Maps.Map(document.getElementById("mapDiv"),
{credentials:"Bing Maps Key", center: new Microsoft.Maps.Location(51.5, -.1), zoom: 10,
mapTypeId:Microsoft.Maps.MapTypeId.collinsBart});
      }
      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative; width:600px; height:600px;"></div>
   </body>
</html>
```

See Also

Changing the Map View

Returning a Localized Map

MouseEventArgs Object

Contains the arguments for mouse events.

Properties

Name	Type	Description
eventName	*string*	The event that occurred.
handled	*boolean*	A boolean indicating whether the event is handled. If this property is set to **true**, the default map control behavior

Name	Type	Description
		for the event is cancelled.
isPrimary	*boolean*	A boolean indicating if the primary button (such as the left mouse button or a tap on a touch screen) was used.
isSecondary	*boolean*	A boolean indicating if the secondary mouse button (such as the right mouse button) was used.
isTouchEvent	*boolean*	A boolean indicating whether the event that occurred was a touch event.
originalEvent	*object*	The original browser event.
pageX	*number*	The x-value of the pixel coordinate on the page of the mouse cursor.
pageY	*number*	The y-value of the pixel coordinate on the page of the mouse cursor.
target	*object*	The object that fired the event.
targetType	*string*	The type of the object that fired the event. Valid values include the following: ‘map’, ‘polygon’, ‘polyline’, or ‘pushpin’
wheelDelta	*number*	The number of units that the mouse wheel has changed.

Methods

Name	Definition	Return Value	Description
getX	`getX()`	*number*	Returns the x-value of the pixel coordinate, relative to the map, of the mouse.

Name	Definition	Return Value	Description
getY	`getY()`	*number*	Returns the y-value of the pixel coordinate, relative to the map, of the mouse.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         var map = null;

         function GetMap()
         {
            // Initialize the map
            map = new Microsoft.Maps.Map(document.getElementById("myMap"),
                         {credentials:"Bing Maps Key"});

            //Add handler for the map click event.
            Microsoft.Maps.Events.addHandler(map, 'click', displayEventInfo);

         }

          function displayEventInfo(e) {
            if (e.targetType == "map") {
                var point = new Microsoft.Maps.Point(e.getX(), e.getY());
                var loc = e.target.tryPixelToLocation(point);
                document.getElementById("textBox").value = loc.latitude + ", " +
loc.longitude;

            }
        }

      </script>
   </head>
   <body onload="GetMap();">
      <div id='myMap' style="position:relative; width:400px; height:400px;"></div><br>
      <b>Click the map to display the coordinate values at that point.</b><br>
      Latitude, Longitude:
      <input id='textBox' type="text" style="width:250px;"/>
   </body>
</html>
```

PixelReference Enumeration

Contains constants used to show how pixels are defined.

Constants

Name	Description
control	The pixel is defined relative to the map control's root element, where the top left corner of the map control is (0, 0). Using this option might cause a page reflow which may negatively impact performance.
page	The pixel is defined relative to the page, where the top left corner of the HTML page is (0, 0). This option is best used when working with mouse or touch events. Using this option might

Name	Description
	cause a page reflow which may negatively impact performance.
viewport	The pixel is defined in viewport coordinates, relative to the center of the map, where the center of the map is (0, 0). This option is best used for positioning geo-aligned entities added to the user layer.

Methods

Name	Definition	Return Value	Description
isValid	`isValid(reference:`PixelReference`)`	*boolean*	Determines whether the specified `reference` is a supported PixelReference.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

   <head>

      <title></title>

      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">


      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>


      <script type="text/javascript">


         var map = null;


         function GetMap()
         {
            // Initialize the map
            map = new Microsoft.Maps.Map(document.getElementById("myMap"),
                         {credentials:"Bing Maps Key"});

            // Add handler for the map click event.
            Microsoft.Maps.Events.addHandler(map, 'click', displayEventInfo);


         }

          function displayEventInfo(e) {
              if (e.targetType == "map") {

                  var point = new Microsoft.Maps.Point(e.pageX, e.pageY,
Microsoft.Maps.PixelReference.page);
                  var loc = e.target.tryPixelToLocation(point,
Microsoft.Maps.PixelReference.page);

                  if (loc!=null)
                  {
                     alert("The location " + loc.latitude + ", " + loc.longitude + " was
clicked.");
                  }

              }
          }



      </script>
   </head>
   <body onload="GetMap();">
      <div id='myMap' style="position:relative; width:400px; height:400px;"></div><br>
      <b>Click the map to display the coordinate values at that point.</b><br>
   </body>
</html>
```

Point Class

Represents a point on the map.

Constructor

Name	Definition	Description
Point	`Point(x:`*number*`, y:`*number*`)`	Initializes a new instance of the Point class.

Properties

Name	Type	Description
x	*number*	The x value of the coordinate.
y	*number*	The y-value of the coordinate.

Static Methods

Name	Definition	Return Value	Description
areEqual	`areEqual(point1:`Point`, point2:`Point`)`	*boolean*	Determines if the specified points are equal.
clone	`clone(`Point`)`	Point	Returns a copy of the Point object.

Methods

Name	Definition	Return Value	Description
clone	`clone()`	Point	Returns a copy of the Point object.
toString	`toString()`	*string*	Converts the Point object into a string.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         var map = null;

         function GetMap()
         {
            // Initialize the map
            map = new Microsoft.Maps.Map(document.getElementById("myMap"),
                         {credentials:"Bing Maps Key "});

            // Add handler for the map click event.
            Microsoft.Maps.Events.addHandler(map, 'click', displayEventInfo);



         }
```

```
          function displayEventInfo(e) {
              if (e.targetType == "map") {

                  var point = new Microsoft.Maps.Point(e.getX(), e.getY());
                  var loc = e.target.tryPixelToLocation(point);

                  if (loc!=null)
                  {
                     alert("The location " + loc.latitude + ", " + loc.longitude + " was
clicked.");
                  }

              }
          }

      </script>
   </head>
   <body onload="GetMap();">
      <div id='myMap' style="position:relative; width:400px; height:400px;"></div><br>
      <b>Click the map to display the coordinate values at that point.</b><br>
   </body>
</html>
```

Polygon Class (AJAX)

Represents a polygon on the map.

Constructor

Name	Definition	Description
Polygon	**Polygon**(locations:Location*[]*, options?:PolygonOptions)	Initializes a new instance of the Polygon class.

Methods

Name	Definition	Return Value	Description
getFillColor	`getFillColor()`	Color	Returns the color of the inside of the polygon.
getLocations	`getLocations()`	Location*[]*	Returns the locations that define the corners of the polygon.
getStrokeColor	`getStrokeColor()`	Color	Returns the color of the outline of the polygon.
getStrokeThickness	`getStrokeThickness()`	*number*	Returns the thickness of the outline of the polygon.
getVisible	`getVisible()`	*boolean*	Returns whether the polygon is visible. A value of **false** indicates that the polygon is hidden, although it is still an entity on the map.
setLocations	`setLocations(locations:`Location*[]*`)`	None	Sets the locations that define the corners of the polygon.
setOptions	`setOptions(options:`PolygonOptions`)`	None	Sets options for the polygon.

Name	Definition	Return Value	Description
toString	`toString()`	*string*	Converts the Polygon object to a string.

Events

Name	Arguments	Description
click	*eventArgs*:MouseEventArgs	Occurs when the mouse is used to click the polygon.
dblclick	*eventArgs*:MouseEventArgs	Occurs when the mouse is used to double click the polygon.
entitychanged	*object*: {entity:*Entity*}	Occurs when the location of the polygon or any of the polygon's options change.
mousedown	*eventArgs*:MouseEventArgs	Occurs when the left mouse button is pressed when the mouse is over the polygon.
mouseout	*eventArgs*:MouseEventArgs	Occurs when the mouse cursor moves out of the area covered by the polygon.
mouseover	*eventArgs*:MouseEventArgs	Occurs when the mouse is over the polygon.
mouseup	*eventArgs*:MouseEventArgs	Occurs when the left mouse button is lifted up when the mouse is over the polygon.
rightclick	*eventArgs*:MouseEventArgs	Occurs when the right mouse button is used to click the polygon.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         function GetMap()
         {
            // Initialize the map
            var map = new Microsoft.Maps.Map(document.getElementById("mapDiv"),
{credentials:"Bing Maps Key"});

            // Create the locations
            var location1 = new Microsoft.Maps.Location(20,-20);
            var location2 = new Microsoft.Maps.Location(20,20);
            var location3 = new Microsoft.Maps.Location(-20,20);
            var location4 = new Microsoft.Maps.Location(-20,-20);



            // Create a polygon
            var vertices = new Array(location1, location2, location3, location4,
location1);
            var polygoncolor = new Microsoft.Maps.Color(100,100,0,100);
            var polygon = new Microsoft.Maps.Polygon(vertices,{fillColor: polygoncolor,
strokeColor: polygoncolor});

            // Add the shape to the map
            map.entities.push(polygon)

            // Set the view
            map.setView({bounds: Microsoft.Maps.LocationRect.fromLocations(vertices)});

         }

      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>
   </body>
</html>
```

PolygonOptions Object

Represents the options for a polygon.

Properties

Name	Type	Description
fillColor	Color	The color of the inside of the polygon.
strokeColor	Color	The color of the outline of the polygon.
strokeThickness	*number*	The thickness of the outline of the polygon.
visible	*boolean*	A boolean indicating whether to show or hide the polygon. A value of **false** indicates that the polygon is hidden, although it is still an entity on the map.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         function GetMap()
         {
            // Initialize the map
            var map = new Microsoft.Maps.Map(document.getElementById("mapDiv"),
{credentials:"Bing Maps Key"});

            // Create the locations
            var location1 = new Microsoft.Maps.Location(20,-20);
            var location2 = new Microsoft.Maps.Location(20,20);
            var location3 = new Microsoft.Maps.Location(-20,20);
            var location4 = new Microsoft.Maps.Location(-20,-20);



            // Create a polygon
            var vertices = new Array(location1, location2, location3, location4,
location1);
            var polygon = new Microsoft.Maps.Polygon(vertices,
                              {fillColor: new Microsoft.Maps.Color(100,100,0,100),
                               strokeColor: new Microsoft.Maps.Color(200,0,100,100),
                               strokeThickness: 5});
            // Add the shape to the map
            map.entities.push(polygon)

            // Set the view
            map.setView({bounds: Microsoft.Maps.LocationRect.fromLocations(vertices)});

         }



      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>
   </body>
</html>
```

Polyline Class

Represents a polyline on the map.

Constructor

Name	Definition	Description
Polyline	`Polyline(locations:`Location*[]*, `options?:`PolylineOptions`)`	Initializes a new instance of the Polyline class.

Methods

Name	Definition	Return Value	Description
getLocations	`getLocations()`	Location*[]*	Returns the locations that define the

Name	Definition	Return Value	Description
			polyline.
getStrokeColor	`getStrokeColor()`	Color	Returns the color of the polyline.
getStrokeThickness	`getStrokeThickness()`	*number*	Returns the thickness of the polyline.
getVisible	`getVisible()`	*boolean*	Returns whether the polyline is visible. A value of **false** indicates that the polyline is hidden, although it is still an entity on the map.
setLocations	`setLocations(locations:`Location*[]*`)`	None	Sets the locations that define the polyline.
setOptions	`setOptions(options:`PolylineOptions`)`	None	Sets options for the polyline.
toString	`toString()`	*string*	Converts the Polyline object to a string.

Events

Name	Arguments	Description
click	*eventArgs:*MouseEventArgs	Occurs when the mouse is used to click the polyline.
dblclick	*eventArgs:*MouseEventArgs	Occurs when the mouse is used to double click the

Name	Arguments	Description
		polyline.
entitychanged	*object*: {entity:*Entity*}	Occurs when the location of the polyline or any of the polyline’s options change.
mousedown	*eventArgs*:MouseEventArgs	Occurs when the left mouse button is pressed when the mouse is over the polyline.
mouseout	*eventArgs*:MouseEventArgs	Occurs when the mouse cursor moves out of the area covered by the polyline.
mouseover	*eventArgs*:MouseEventArgs	Occurs when the mouse is over the polyline.
mouseup	*eventArgs*:MouseEventArgs	Occurs when the left mouse button is lifted up when the mouse is over the polyline.
rightclick	*eventArgs*:MouseEventArgs	Occurs when the right mouse button is used to click the polyline.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

   <head>

      <title></title>

      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">


      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>


      <script type="text/javascript">
```

```
          var map = null;



         function GetMap()

         {

            // Initialize the map

            map = new
Microsoft.Maps.Map(document.getElementById("mapDiv"),{credentials:"Your Bing Maps Key"});


            // Create the locations

            var location1 = new Microsoft.Maps.Location(-20,-20);

            var location2 = new Microsoft.Maps.Location(20,-20);

            var location3 = new Microsoft.Maps.Location(20,20);

            var location4 = new Microsoft.Maps.Location(60, 20);

            var location5 = new Microsoft.Maps.Location(60, 60);



            // Create a polyline

            var lineVertices = new Array(location1, location2, location3, location4,
location5);

            var line = new Microsoft.Maps.Polyline(lineVertices);


            // Add the polyline to the map

            map.entities.push(line);


         }



      </script>

   </head>

   <body onload="GetMap();">

      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>

   </body>
```

```
</html>
```

PolylineOptions Object

Represents the options for a polyline.

Properties

Name	Type	Description
strokeColor	Color	The color of the polyline.
strokeThickness	*number*	The thickness of the polyline.
visible	*boolean*	A boolean indicating whether to show or hide the polyline. A value of **false** indicates that the polyline is hidden, although it is still an entity on the map.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

          var map = null;
```

```
         function GetMap()
         {
            // Initialize the map
            map = new
Microsoft.Maps.Map(document.getElementById("mapDiv"),{credentials:"Your Bing Maps Key"});

            // Create the locations
            var location1 = new Microsoft.Maps.Location(-20,-20);
            var location2 = new Microsoft.Maps.Location(20,-20);
            var location3 = new Microsoft.Maps.Location(20,20);
            var location4 = new Microsoft.Maps.Location(60, 20);
            var location5 = new Microsoft.Maps.Location(60, 60);



            // Create a polyline
            var lineVertices = new Array(location1, location2, location3, location4,
location5);
            var line = new Microsoft.Maps.Polyline(lineVertices,{strokeColor:new
Microsoft.Maps.Color(200, 100, 0, 100), strokeThickness:10});

            // Add the polyline to the map
            map.entities.push(line);

         }



      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>
   </body>
</html>
```

Pushpin Class (AJAX)

Defines a pushpin on the map.

Constructor

Name	Definition	Description
Pushpin	`Pushpin(location:`Location`, options?:`PushpinOptions`)`	Initializes a new instance of the Pushpin class.

Methods

Name	Definition	Return Value	Description
getAnchor	`getAnchor()`	Point	Returns the point on the pushpin icon which is anchored to the pushpin location. An anchor of (0,0) is the top left corner of the icon.
getIcon	`getIcon()`	*string*	Returns the pushpin icon.
getHeight	`getHeight()`	*number*	Returns the height of the pushpin, which is the height of the pushpin icon.
getLocation	`getLocation()`	Location	Returns the location of the pushpin.
getText	`getText()`	*string*	Returns the text associated with the pushpin.

Name	Definition	Return Value	Description
getTextOffset	`getTextOffset()`	Point	Returns the amount the text is shifted from the pushpin icon.
getTypeName	`getTypeName()`	*string*	Returns the type of the pushpin.
getVisible	`getVisible()`	*boolean*	Returns whether the pushpin is visible. A value of **false** indicates that the pushpin is hidden, although it is still an entity on the map.
getWidth	`getWidth()`	*number*	Returns the width of the pushpin, which is the width of the pushpin icon.
getZIndex	`getZIndex()`	*number*	Returns the z-index of the pushpin with respect to other items on the map.
setLocation	`setLocation(location:Location)`	None	Sets the location of the pushpin.
setOptions	`setOptions(options:PushpinOptions)`	None	Sets options for the pushpin.
toString	`toString()`	*string*	Converts the Pushpin object to a string.

Events

Name	Arguments	Description
click	*eventArgs*:MouseEventArgs	Occurs when the mouse is used to click the pushpin.
dblclick	*eventArgs*:MouseEventArgs	Occurs when the mouse is used to double click the pushpin.
entitychanged	*object*: {entity:*Entity*}	Occurs when the location of the pushpin or any of the pushpin's options change.
mousedown	*eventArgs*:MouseEventArgs	Occurs when the left mouse button is pressed when the mouse is over the pushpin.
mouseout	*eventArgs*:MouseEventArgs	Occurs when the mouse cursor moves out of the area covered by the pushpin.
mouseover	*eventArgs*:MouseEventArgs	Occurs when the mouse is over the pushpin.
mouseup	*eventArgs*:MouseEventArgs	Occurs when the left mouse button is lifted up when the mouse is over the pushpin.
rightclick	*eventArgs*:MouseEventArgs	Occurs when the right mouse button is used to click the pushpin.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

   <head>

      <title></title>

      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         var map = null;

         function GetMap()
         {
            // Initialize the map
            map = new Microsoft.Maps.Map(document.getElementById("myMap"),
                         {credentials:"Bing Maps Key"});

            // Retrieve the location of the map center
            var center = map.getCenter();

            // Add a pin to the center of the map
            var pin = new Microsoft.Maps.Pushpin(center, {draggable: true});

            // Add a handler to the pushpin drag
            Microsoft.Maps.Events.addHandler(pin, 'mouseup', DisplayLoc);

            map.entities.push(pin);
         }

         function DisplayLoc(e){
            if (e.targetType == 'pushpin'){

               var pinLoc = e.target.getLocation();
               alert("The location of the pushpin is now " + pinLoc.latitude + ", " +
pinLoc.longitude);

            }
}
```

```
      </script>
   </head>
   <body onload="GetMap();">
      <div id='myMap' style="position:relative; width:400px; height:400px;"></div>
      <b>Drag the pushpin to a new location.</b>
   </body>
</html>
```

PushpinOptions Object

Represents the options for a pushpin.

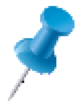
Properties

Name	Type	Description
anchor	Point	The point on the pushpin icon which is anchored to the pushpin location. An anchor of (0,0) is the top left corner of the icon. The default anchor is the bottom center of the icon.
draggable	*boolean*	A boolean indicating whether the pushpin can be dragged to a new position with the mouse.
icon	*string*	The path of the image to use as the pushpin icon.
height	*number*	The height of the pushpin, which is the height of the pushpin icon. The default value is 39.
text	*string*	The text associated with the pushpin.
textOffset	Point	The amount the text is shifted from the pushpin icon. The default value is (0,5).

Name	Type	Description
typeName	*string*	The type of the pushpin, as a string. The pushpin DOM (document object model) node created for the pushpin will have the specified typeName.
visible	*boolean*	A boolean indicating whether to show or hide the pushpin. The default value is **true**. A value of **false** indicates that the pushpin is hidden, although it is still an entity on the map.
width	*number*	The width of the pushpin, which is the width of the pushpin icon. The default value is 25.
zIndex	*number*	The z-index of the pushpin with respect to other items on the map.

Example

This example uses the image below, named “BluePushpin.png”, as the pushpin icon.



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">
```

```
         var map = null;

         function GetMap()
         {
            // Initialize the map
            map = new Microsoft.Maps.Map(document.getElementById("myMap"),
                         {credentials:"Bing Maps Key"});

            // Retrieve the location of the map center
            var center = map.getCenter();

            // Add a pin to the center of the map
            var pin = new Microsoft.Maps.Pushpin(center, {icon:"BluePushpin.png",
height:50, width:50, anchor:new
Microsoft.Maps.Point(0,50), draggable: true});



            map.entities.push(pin);
         }

      </script>
   </head>
   <body onload="GetMap();">
      <div id='myMap' style="position:relative; width:400px; height:400px;"></div>

   </body>
</html>
```

TileLayer Class

Represents a tile layer.

Constructor

Name	Definition	Description
TileLayer	`TileLayer(options:`TileLayerOptions`)`	Initializes a new instance of the TileLayer class.

Methods

Name	Definition	Return Type	Description
getOpacity	`getOpacty()`	*number*	Returns the opacity of the tile layer, defined as a double between 0 (not visible) and 1.
getTileSource	`getTileSource(projection:`*string*`)`	TileSource	Returns the tile source of the tile layer. The `projection` parameter accepts the following values: *mercator*, *enhancedBirdseyeNorthUp*, *enhancedBirdseyeSouthUp*, *enhancedBirdseyeEastUp*, *enhancedBirdseyeWestUp*
getZIndex	`getZIndex()`	*number*	Returns the z-index of the tile layer with respect to other items on the map.
setOptions	`setOptions(options:`TileLayerOptions`)`	None	Sets options for the tile layer.
toString	`toString()`	*string*	Converts the TileLayer object to a string.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         function GetMap()
         {
            // Initialize the map
            var map = new Microsoft.Maps.Map(document.getElementById("mapDiv"),
{credentials:"Bing Maps Key", center:new Microsoft.Maps.Location(48.03,-
122.4), zoom:12, mapTypeId:"r"});

            try
            {
               // Create the tile layer source
               var tileSource = new Microsoft.Maps.TileSource({uriConstructor:
'http://www.microsoft.com/maps/isdk/ajax/layers/lidar/{quadkey}.png'});

               // Construct the layer using the tile source
               var tilelayer= new Microsoft.Maps.TileLayer({ mercator: tileSource,
opacity: .7 });

               // Push the tile layer to the map
               map.entities.push(tilelayer);

            }
            catch(err)
            {
               alert( 'Error Message:' + err.message);
```

```
                }
            }
        </script>
    </head>
    <body onload="GetMap();">
        <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>
    </body>
</html>
```

See Also

Working with Tile Layers

TileLayerOptions Object

Defines the options for a tile layer.

Properties

Name	Type	Description
mercator	TileSource	The tile source for the tile layer.
opacity	*number*	The opacity of the tile layer, defined by a number between 0 (not visible) and 1.
visible	*boolean*	A boolean indicating whether to show or hide the tile layer. The default value is **true**. A value of **false** indicates that the tile layer is hidden, although it is still an entity on the map.
zIndex	*number*	The z-index of the tile layer, with respect to other items on

Name	Type	Description
		the map.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         var map = null;
         var tilelayer = null;

         function GetMap()
         {
            // Initialize the map
            map = new Microsoft.Maps.Map(document.getElementById("mapDiv"),
{credentials:"Bing Maps Key", center:new Microsoft.Maps.Location(48.03,-
122.4), zoom:12, mapTypeId:"r"});

            try
            {
               // Create the tile layer source
               var tileSource = new Microsoft.Maps.TileSource({uriConstructor:
'http://www.microsoft.com/maps/isdk/ajax/layers/lidar/{quadkey}.png'});

               // Construct the layer using the tile source
               tilelayer= new Microsoft.Maps.TileLayer({ mercator: tileSource, opacity:
.7 });

               // Push the tile layer to the map
               map.entities.push(tilelayer);

            }
            catch(err)
            {
               alert( 'Error Message:' + err.message);
            }


         }

         function SetOpacity()
        {

           var opacityVal = parseFloat(document.getElementById("txtOpacity").value);

           if ((opacityVal > 1) || (opacityVal < 0))
           {
              alert("The opacity value must be between 0 and 1.");
           }
           else
           {
              tilelayer.setOptions({opacity: opacityVal});
           }

        }


      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>
<input id="txtOpacity" type="text" value=".1" style="width:25px;"/>
<input type="button" value="Set Opacity" onclick="SetOpacity();"/>
   </body>
</html>
```

See Also

Working with Tile Layers

TileSource Class

Defines a tile source for a tile layer.

Constructor

Name	Definition	Description
TileSource	`TileSource(options:`TileSourceOptions`)`	Initializes a new instance of the TileSource class.

Methods

Name	Definition	Return Type	Description
getHeight	`getHeight()`	*Number*	Returns the pixel height of each tile in the tile source.
getUriConstructor	`getUriConstructor()`	*string*	Returns a string that constructs tile URLs used to retrieve tiles for the tile layer.
getWidth	`getWidth()`	*number*	Returns the pixel width of each tile in the tile source.

Name	Definition	Return Type	Description
toString	`toString()`	*string*	Converts the TileSource object to a string.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         function GetMap()
         {
            // Initialize the map
            var map = new
Microsoft.Maps.Map(document.getElementById("mapDiv"),{credentials:"Bing Maps Key",
center:new Microsoft.Maps.Location(48.03,-122.4), zoom:12, mapTypeId:"r"});

            try
            {
               // Create the tile layer source
               var tileSource = new Microsoft.Maps.TileSource({uriConstructor:
'http://www.microsoft.com/maps/isdk/ajax/layers/lidar/{quadkey}.png'});

               // Construct the layer using the tile source
               var tilelayer= new Microsoft.Maps.TileLayer({ mercator: tileSource,
opacity: .7 });
```

```
                // Push the tile layer to the map
                map.entities.push(tilelayer);

             }
             catch(err)
             {
                alert( 'Error Message:' + err.message);
             }



          }



       </script>
    </head>
    <body onload="GetMap();">
       <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>
    </body>
</html>
```

See Also

Working with Tile Layers

TileSourceOptions Object

Defines options for a tile source.

Properties

Name	Type	Description
height	*number*	The pixel height of each tile in the tile source. The default value is 256.

Name	Type	Description
		The specified height needs to be a multiplier of 2 of the current projection’s tile height for the tiles to be shown. For example, since Mercator tile source tiles are 256x256, this projection supports tiles that are 64x64, 128x128, 256x256, 512x512, or any combination of these.
uriConstructor	*string*	The string that constructs the URLs used to retrieve tiles from the tile source. This property is required. The uriConstructor will replace {subdomain} and {quadkey}.
width	*number*	The pixel width of each tile in the tile source. The default value is 256. The specified width needs to be a multiplier of 2 of the current projection’s tile width for the tiles to be shown. For example, since Mercator tile source tiles are 256x256, this projection supports tiles that are 64x64, 128x128, 256x256, 512x512, or any combination of these.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         function GetMap()
         {
            // Initialize the map
            var map = new
Microsoft.Maps.Map(document.getElementById("mapDiv"),{credentials:"Bing Maps Key",
center:new Microsoft.Maps.Location(48.03,-122.4), zoom:12, mapTypeId:"r"});

            try
            {
               // Create the tile layer source
               var tileSource = new Microsoft.Maps.TileSource({uriConstructor:
'http://www.microsoft.com/maps/isdk/ajax/layers/lidar/{quadkey}.png'});

               // Construct the layer using the tile source
               var tilelayer= new Microsoft.Maps.TileLayer({ mercator: tileSource,
opacity: .7 });

               // Push the tile layer to the map
               map.entities.push(tilelayer);

            }
            catch(err)
            {
               alert( 'Error Message:' + err.message);
            }

         }

      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative; width:400px; height:400px;"></div>
   </body>
</html>
```

See Also

Working with Tile Layers

ViewOptions Object

Contains options for the map view.

Properties

Name	Type	Description
animate	*boolean*	A boolean that specifies whether to animate map navigation. Note that this option is associated with each setView call and defaults to true if not specified.
bounds	LocationRect	The bounding rectangle of the map view.
center	Location	The location of the center of the map view.
centerOffset	Point	The amount the center is shifted.
heading	*number*	The directional heading of the map. The heading is represented in geometric degrees with 0 or 360 = North, 90 = East, 180 = South, and 270 = West.
labelOverlay	LabelOverlay	A constant indicating how map labels are displayed.

Name	Type	Description
mapTypeId	*string*	The map type of the view. Valid map types are found in the MapTypeId Enumeration topic.
padding	*number*	The amount of padding to be added to each side of the bounds of the map view.
zoom	*number*	The zoom level of the map view.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

      <script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>

      <script type="text/javascript">

         var map = null;

         function GetMap()
         {
            // Set the initial map and view settings

            var initialViewBounds = Microsoft.Maps.LocationRect.fromCorners(new
Microsoft.Maps.Location(43,-123), new Microsoft.Maps.Location(33,-113));
            var options = {credentials:"Bing Maps Key", width: 500, height: 500, bounds:
initialViewBounds, mapTypeId:Microsoft.Maps.MapTypeId.aerial, animate: false};
```

```
            // Initialize the map
            map = new Microsoft.Maps.Map(document.getElementById("mapDiv"),options);



         }


         function SetZoom()
         {
            // Retrieve the zoom level set by the user - converting it to a number.
            var zoomLevel = parseInt(document.getElementById("txtZoom").value);


            // Get the existing options.
            var options = map.getOptions();


            // Set the zoom level of the map
            options.zoom = zoomLevel;
            map.setView(options);


         }


      </script>
   </head>
   <body onload="GetMap();">
      <div id='mapDiv' style="position:relative;"></div>
      <input id="txtZoom" type="text" value="1"/>
      <input type="button" value="Set Zoom" onclick="SetZoom();"/>
   </body>
</html>
```

Bing Maps AJAX Control 7.0 Supported Browsers

This topic contains information about browser support for the Bing Maps AJAX Control 7.0.

> The Bing Maps AJAX Control 7.0 uses features of HTML5 if it detects that the client browser supports HTML5. If this is the case, map performance will be faster, and map animations and transitions will be smoother.

Supported Browsers

The Bing Maps AJAX Control 7.0 is supported on the following Web browsers. If you are not using a supported Web browser, certain features of the map control may not work.

Desktop Browser	Description
Internet Explorer 7.0	Supported on the PC
Internet Explorer 8.0	Supported on the PC
Internet Explorer 9.0	Supported on the PC
Firefox 3.6	Supported on the PC and the Mac
Firefox 4.0	Supported on the PC and the Mac
Safari 5	Supported on the Mac
Google Chrome	Supported on the PC

Mobile Browser
Apple 3GS/4.0 iPhone Browser
Google Android 2.X Browser
Research in Motion (RIM) BlackBerry 6.0 Browser

Bing Maps AJAX Control 7.0 Developer Resources

This topic contains support resources and contact information.

Developer Resources

The following resources are available for Bing Maps developers:

- Connect with other Bing Maps developers on the Bing Maps AJAX Control Forum.
- Visit the http://www.microsoft.com/maps website.
- Read the Bing Maps Developer blog

Account Issues

If you are having issues creating a Bing Maps Developer Account, getting a Bing Maps Key, or have an account access question, contact mpnet@microsoft.com.

Licensing Questions

If you are interested in finding out more about Bing Maps or have questions about licensing Bing Maps, email maplic@microsoft.com or go to http://www.microsoft.com/maps/resources/default.aspx. From North, Central, and South America, you can also contact Bing Maps by calling (800) 426-9400, ext. 11315.