

Street View Service

Overview

Google Street View provides panoramic 360 degree views from designated roads throughout its coverage area. Street View's API coverage is the same as that for the Google Maps application (<https://maps.google.com/>). The list of currently supported cities for Street View is available at the [Google Maps website](#) (<http://www.google.com/maps/about/behind-the-scenes/streetview/#where>).

A sample Street View image is shown below.



The Google Maps JavaScript API provides a Street View service for obtaining and manipulating the imagery used in Google Maps Street View. This Street View service is supported natively within the browser.

Street View Map Usage

Although Street View can be used within a [standalone DOM element](#) (#StreetViewContainers), it is most useful when indicating a location on a map. By default, Street View is enabled on a map, and a Street View *Pegman control* appears integrated within the navigation (zoom and pan) controls. You may hide this control within the map's `MapOptions` by setting `streetViewControl` to `false`. You may also change the default position of the Street View control by setting the Map's `streetViewControlOptions.position` property to a new `ControlPosition`.

The Street View Pegman control allows you to view Street View panoramas directly within the map. When the user clicks and holds the Pegman, the map updates to show blue outlines around Street View-enabled streets, offering a user experience similar to the Google Maps app.

When the user drops the Pegman marker onto a street, the map updates to display a Street View panorama of the indicated location.

Street View Panoramas

Street View images are supported through use of the `StreetViewPanorama` object, which provides an API interface to a Street View "viewer." Each map contains a default Street View panorama, which you can retrieve by calling the map's `getStreetView()` method. When you add a Street View control to the map by setting its `streetViewControl` option to `true`, you automatically connect the Pegman control to this default Street View panorama.

You may also create your own `StreetViewPanorama` object and set the map to use that instead of the default, by setting the map's `streetView` property explicitly to that constructed object. You may wish to override the default panorama if you want to modify default behavior, such as the automatic sharing of overlays between the map and the panorama. (See [Overlays within Street View](#) (#StreetViewOverlays) below.)

Street View Containers

You may instead wish to display a `StreetViewPanorama` within a separate DOM element, often a `<div>` element. Simply pass the DOM element within the `StreetViewPanorama`'s constructor. For optimum display of images, we recommend a minimum size of

200 pixels by 200 pixels.

Note: although Street View functionality is designed to be used in conjunction with a map, this usage is not required. You may use a standalone Street View object without a map.

Street View Locations and Point-of-View (POV)

The `StreetViewPanorama` constructor also allows you to set the Street View location and point of view using the `StreetViewOptions` parameter. You may call `setPosition()` and `setPov()` on the object after construction to change its location and POV.

The Street View location defines the placement of the camera focus for an image, but it does not define the orientation of the camera for that image. For that purpose, the `StreetViewPov` object defines two properties:

- `heading` (default 0) defines the rotation angle around the camera locus in degrees relative from true north. Headings are measured clockwise (90 degrees is true east).
- `pitch` (default 0) defines the angle variance "up" or "down" from the camera's initial default pitch, which is often (but not always) flat horizontal. (For example, an image taken on a hill will likely exhibit a default pitch that is not horizontal.) Pitch angles are measured with positive values looking up (to +90 degrees straight up and orthogonal to the default pitch) and negative values looking down (to -90 degrees straight down and orthogonal to the default pitch).

The `StreetViewPov` object is most often used to determine the point of view of the Street View camera. You can also determine the point-of-view of the photographer — typically the direction the [car or trike](#) (<https://maps.google.com/help/maps/streetview/learn/cars-trikes-and-more.html>) was facing — with the `StreetViewPanorama.getPhotographerPov()` method.

The following code displays a map of Boston with an initial view of Fenway Park. Selecting the Pegman and dragging it to a supported location on the map will change the Street View panorama:

```
function initialize() {
  var fenway = {lat: 42.345573, lng: -71.098326};
  var map = new google.maps.Map(document.getElementById('map'), {
    center: fenway,
    zoom: 14
  });
  var panorama = new google.maps.StreetViewPanorama(
    document.getElementById('pano'), {
      position: fenway,
      pov: {
        heading: 34,
        pitch: 10
      }
    });
  map.setStreetView(panorama);
}
```

[[view example](#)] (<https://developers.google.com/maps/documentation/javascript/examples/streetview-simple>)

Motion tracking on mobile devices

On devices that support device orientation events, the API offers users the ability to change the Street View point of view based on the movement of the device. Users can look around by moving their devices. This is called motion tracking or device rotation tracking.

As app developer, you can change the default behavior as follows:

- Enable or disable the motion tracking functionality. By default, motion tracking is enabled on any device that supports it. The following sample disables motion tracking, but leaves the motion tracking control visible. (Note that the user can turn on motion tracking by tapping the control.)

```
var panorama = new google.maps.StreetViewPanorama(
  document.getElementById('pano'), {
    position: {lat: 37.869260, lng: -122.254811},
    pov: {heading: 165, pitch: 0},
```

```
    motionTracking: false
});
```

- Hide or show the motion tracking control. By default, the control is shown on devices that support motion tracking. The user can tap the control to turn motion tracking on or off. Note that the control will never appear if the device doesn't support motion tracking, regardless of the value of `motionTrackingControl`.

The following sample disables both motion tracking and the motion tracking control. In this case, the user can't turn motion tracking on:

```
var panorama = new google.maps.StreetViewPanorama(
  document.getElementById('pano'), {
    position: {lat: 37.869260, lng: -122.254811},
    pov: {heading: 165, pitch: 0},
    motionTracking: false,
    motionTrackingControl: false
});
```

- Change the default position of the motion tracking control. By default, the control appears near the bottom right of the panorama (position `RIGHT_BOTTOM`). The following sample sets the position of the control to left bottom:

```
var panorama = new google.maps.StreetViewPanorama(
  document.getElementById('pano'), {
    position: {lat: 37.869260, lng: -122.254811},
    pov: {heading: 165, pitch: 0},
    motionTrackingControlOptions: {
      position: google.maps.ControlPosition.LEFT_BOTTOM
    }
});
```

To see motion tracking in action, view the following sample on a mobile device (or any device that supports device orientation events):

2692 Bancroft Way
Berkeley, California
[View on Google Maps](#)



Google

[Report a problem](#)

[View the example on a new page \(streetview-embed.html\)](#)

(<https://developers.google.com/maps/documentation/javascript/examples/streetview-embed>).

Overlays within Street View

The default `StreetViewPanorama` object supports the native display of map [overlays](#)

(<https://developers.google.com/maps/documentation/javascript/overlays>). Overlays generally appear at "street level" anchored at `LatLng` positions. (Markers will appear with their tails anchored to the location's horizontal plane within the Street View panorama for example.)

Currently, the types of overlays which are supported on Street View panoramas are limited to `Markers`, `InfoWindows` and custom `OverlayViews`. Overlays which you display on a map may be displayed on a Street View panorama by treating the panorama as a substitute for the `Map` object, calling `setMap()` and passing the `StreetViewPanorama` as an argument instead of a map. Info windows similarly may be opened within a Street View panorama by calling `open()`, passing the `StreetViewPanorama()` instead of a map.

Additionally, when creating a map with a default `StreetViewPanorama`, any markers created on a map are shared automatically with the map's associated Street View panorama, provided that panorama is visible. To retrieve the default Street View panorama, call `getStreetView()` on the `Map` object. Note that if you explicitly set the map's `streetView` property to a `StreetViewPanorama` of your own construction, you will override the default panorama and disable automatic overlay sharing.

The following example shows markers denoting various locations around Astor Place, New York City. Toggle the display to Street View to show the shared markers displaying within the `StreetViewPanorama`.

```
var panorama;

function initMap() {
  var astorPlace = {lat: 40.729884, lng: -73.990988};

  // Set up the map
  var map = new google.maps.Map(document.getElementById('map'), {
    center: astorPlace,
    zoom: 18,
    streetViewControl: false
  });

  // Set up the markers on the map
  var cafeMarker = new google.maps.Marker({
    position: {lat: 40.730031, lng: -73.991428},
    map: map,
    icon: 'https://chart.apis.google.com/chart?chst=d_map_pin_icon&chld=cafe|FFFF00',
    title: 'Cafe'
  });

  var bankMarker = new google.maps.Marker({
    position: {lat: 40.729681, lng: -73.991138},
    map: map,
    icon: 'https://chart.apis.google.com/chart?chst=d_map_pin_icon&chld=dollar|FFFF00',
    title: 'Bank'
  });

  var busMarker = new google.maps.Marker({
    position: {lat: 40.729559, lng: -73.990741},
    map: map,
    icon: 'https://chart.apis.google.com/chart?chst=d_map_pin_icon&chld=bus|FFFF00',
    title: 'Bus Stop'
  });

  // We get the map's default panorama and set up some defaults.
  // Note that we don't yet set it visible.
  panorama = map.getStreetView();
  panorama.setPosition(astorPlace);
  panorama.setPov({/** @type {google.maps.StreetViewPov} */{
    heading: 265,
    pitch: 0
  }});
}

function toggleStreetView() {
  var toggle = panorama.getVisible();
  if (toggle == false) {
    panorama.setVisible(true);
  } else {
    panorama.setVisible(false);
  }
}
```

[\[view example\]](https://developers.google.com/maps/documentation/javascript/examples/streetview-overlays) (<https://developers.google.com/maps/documentation/javascript/examples/streetview-overlays>)

Street View Events

When navigating between Street View or manipulating its orientation, you may wish to monitor several events that indicate changes to the `StreetViewPanorama`'s state:

- `pano_changed` fires whenever the individual pano ID changes. This event does not guarantee that any associated data within the panorama (such as the links) has also changed by the time this event is triggered; this event only indicates that a pano ID has changed. Note that the pano ID (which you can use to reference this panorama) is only stable within the current browser session.
- `position_changed` fires whenever the underlying (`LatLng`) position of the panorama changes. Rotating a panorama will not trigger this event. Note that you could change a panorama's underlying position without changing the associated pano

ID, since the API will automatically associate the nearest pano ID to the panorama's position.

- `pov_changed` fires whenever the Street View's `StreetViewPov` changes. Note that this event may fire while the position, and pano ID, remain stable.
- `links_changed` fires whenever the Street View's `links` change. Note that this event may fire asynchronously after a change in the pano ID indicated through `pano_changed`.
- `visible_changed` fires whenever the Street View's visibility changes. Note that this event may fire asynchronously after a change in the pano ID indicated through `pano_changed`.

The following code illustrates how these events can be handled to collect data about the underlying `StreetViewPanorama`:

```
function initPano() {
  var panorama = new google.maps.StreetViewPanorama(
    document.getElementById('pano'), {
      position: {lat: 37.869, lng: -122.255},
      pov: {
        heading: 270,
        pitch: 0
      },
      visible: true
    });

  panorama.addListener('pano_changed', function() {
    var panoCell = document.getElementById('pano-cell');
    panoCell.innerHTML = panorama.getPano();
  });

  panorama.addListener('links_changed', function() {
    var linksTable = document.getElementById('links_table');
    while (linksTable.hasChildNodes()) {
      linksTable.removeChild(linksTable.lastChild);
    }
    var links = panorama.getLinks();
    for (var i in links) {
      var row = document.createElement('tr');
      linksTable.appendChild(row);
      var labelCell = document.createElement('td');
      labelCell.innerHTML = '<b>Link: ' + i + '</b>';
      var valueCell = document.createElement('td');
      valueCell.innerHTML = links[i].description;
      linksTable.appendChild(labelCell);
      linksTable.appendChild(valueCell);
    }
  });

  panorama.addListener('position_changed', function() {
    var positionCell = document.getElementById('position-cell');
    positionCell.firstChild.nodeValue = panorama.getPosition() + '';
  });

  panorama.addListener('pov_changed', function() {
    var headingCell = document.getElementById('heading-cell');
    var pitchCell = document.getElementById('pitch-cell');
    headingCell.firstChild.nodeValue = panorama.getPov().heading + '';
    pitchCell.firstChild.nodeValue = panorama.getPov().pitch + '';
  });
}
```

[\[view example\]](#) (<https://developers.google.com/maps/documentation/javascript/examples/streetview-events>)

Street View Controls

When displaying a `StreetViewPanorama`, a variety of controls appear on the panorama by default. You can enable or disable these controls by setting their appropriate fields within the `StreetViewPanoramaOptions`

(<https://developers.google.com/maps/documentation/javascript/3.exp/reference#StreetViewPanoramaOptions>) to `true` or `false`:

- A `panControl` provides a way to rotate the panorama. This control appears by default as a standard integrated compass and pan control. You may alter the control's position by providing `PanControlOptions` within the `panControlOptions` field.

- A `zoomControl` provides a way to zoom within the image. This control appears by default near the bottom right of the panorama. You can alter the control's appearance by providing `ZoomControlOptions` within the `zoomControlOptions` field.
- An `addressControl` provides a textual overlay indicating the address of the associated location, and offers a link to open the location in Google Maps. You can alter the control's appearance by providing `StreetViewAddressControlOptions` within the `addressControlOptions` field.
- A `fullscreenControl` offers the option to open Street View in fullscreen mode. You can alter the control's appearance by providing `FullscreenControlOptions` within the `fullscreenControlOptions` field.
- A `motionTrackingControl` offers the option to enable or disable motion tracking on mobile devices. This control appears only on devices that support device orientation events. By default, the control appears near the bottom right of the panorama. You can alter the control's position by providing `MotionTrackingControlOptions`. For more information, see the section on [motion tracking](#) (#motion-tracking).
- A `linksControl` provides guide arrows on the image for traveling to adjacent panorama images.
- A Close control allows the user to close the Street View viewer. You can enable or disable the Close control by setting `enableCloseButton` to `true` or `false`.

The following example alters the controls displayed within the associated Street View and removes the view's links:

[View this example full screen](#) (<https://developers.google.com/maps/documentation/javascript/examples/full/streetview-controls>).

```
function initPano() {
  // Note: constructed panorama objects have visible: true
  // set by default.
  var panorama = new google.maps.StreetViewPanorama(
    document.getElementById('map'), {
      position: {lat: 42.345573, lng: -71.098326},
      addressControlOptions: {
        position: google.maps.ControlPosition.BOTTOM_CENTER
      },
      linksControl: false,
      panControl: false,
      enableCloseButton: false
    });
}
```

[\[view example\]](#) (<https://developers.google.com/maps/documentation/javascript/examples/streetview-controls>)

Directly Accessing Street View Data

You may wish to programmatically determine the availability of Street View data, or return information about particular panoramas, without requiring direct manipulation of a map/panorama. You may do so using the `StreetViewService` object, which provides an interface to the data stored in Google's Street View service.

Street View Service Requests

Accessing the Street View service is asynchronous, since the Google Maps API needs to make a call to an external server. For that reason, you need to pass a `callback` method to execute upon completion of the request. This callback method processes the result.

You may initiate two types of requests to the `StreetViewService`:

- Request with a `StreetViewPanoRequest`, this returns panorama data given a reference ID which uniquely identifies the panorama. Note that these reference IDs are only stable for the lifetime of the imagery of that panorama.
- Request with a `StreetViewLocationRequest` this searches for panorama data over a given area, given a passed `LatLng`.

Street View Service Responses

`getPanorama()` needs a `callback` function to execute upon retrieval of a result from the Street View service. This callback function returns a set of panorama data within a `StreetViewPanoramaData` object and a `StreetViewStatus` code denoting the status of the request, in that order.

A `StreetViewPanoramaData` object specification contains meta-data about a Street View panorama of the following form:

```
{
  "location": {
    "latLng": LatLng,
    "description": string,
    "pano": string
  },
  "copyright": string,
  "links": [
    {
      "heading": number,
      "description": string,
      "pano": string,
      "roadColor": string,
      "roadOpacity": number
    }
  ],
  "tiles": {
    "worldSize": Size,
    "tileSize": Size,
    "centerHeading": number
  }
}
```

Note that this data object is not a `StreetViewPanorama` object itself. To create a Street View object using this data, you would need to create a `StreetViewPanorama` and call `setPano()`, passing it the ID as noted in the returned `location.pano` field.

The `status` code may return one of the following values:

- `OK` indicates that the service found a matching panorama.
- `ZERO_RESULTS` indicates that the service could not find a matching panorama with the passed criteria.
- `UNKNOWN_ERROR` indicates that a Street View request could not be processed, though the exact reason is unknown.

The following code creates a `StreetViewService` that responds to user clicks on a map by creating markers which, when clicked, display a `StreetViewPanorama` of that location. The code uses the contents of `StreetViewPanoramaData` returned from the service.

```
/*
 * Click the map to set a new location for the Street View camera.
 */

var map;
var panorama;

function initMap() {
  var berkeley = {lat: 37.869085, lng: -122.254775};
  var sv = new google.maps.StreetViewService();

  panorama = new google.maps.StreetViewPanorama(document.getElementById('pano'));

  // Set up the map.
  map = new google.maps.Map(document.getElementById('map'), {
    center: berkeley,
    zoom: 16,
    streetViewControl: false
  });

  // Set the initial Street View camera to the center of the map
  sv.getPanorama({location: berkeley, radius: 50}, processSVData);

  // Look for a nearby Street View panorama when the map is clicked.
  // getPanoramaByLocation will return the nearest pano when the
  // given radius is 50 meters or less.
  map.addListener('click', function(event) {
    sv.getPanorama({location: event.latLng, radius: 50}, processSVData);
  });
}

function processSVData(data, status) {
  if (status === 'OK') {
    var marker = new google.maps.Marker({
      position: data.location.latLng,
      map: map,
      title: data.location.description
    });
  }
}
```

```
});  
  
panorama.setPano(data.location.pano);  
panorama.setPov({  
  heading: 270,  
  pitch: 0  
});  
panorama.setVisible(true);  
  
marker.addListener('click', function() {  
  var markerPanoID = data.location.pano;  
  // Set the Pano to use the passed panoID.  
  panorama.setPano(markerPanoID);  
  panorama.setPov({  
    heading: 270,  
    pitch: 0  
  });  
  panorama.setVisible(true);  
});  
} else {  
  console.error('Street View data not found for this location.');  
}  
}  
}
```

[\[view example\]](#) (<https://developers.google.com/maps/documentation/javascript/examples/streetview-service>)

Providing Custom Street View Panoramas

The Google Maps JavaScript API supports the display of custom panoramas within the `StreetViewPanorama` object. Using custom panoramas, you can display the interior of buildings, views from scenic locations, or anything from your imagination. You can even link these custom panoramas to Google's existing Street View panoramas.

Setting up a set of custom panorama imagery involves the following steps:

- Create a base panoramic image for each custom panorama. This base image should be at the highest resolution image with which you wish to serve zoomed in imagery.
- (Optional, but recommended) Create a set of panoramic tiles at different zoom levels from the basic image.
- Create links between your custom panoramas.
- (Optional) Designate "entry" panoramas within Google's existing Street View imagery and customize links to/from the custom set to the standard set.
- Define metadata for each panorama image within a `StreetViewPanoramaData` object.
- Implement a method which determines the custom panorama data and images and designate that method as your custom handler within the `StreetViewPanorama` object.

The following sections explain this process.

Creating Custom Panoramas

Each Street View panorama is an image or set of images that provides a full 360 degree view from a single location. The `StreetViewPanorama` object uses images that conform to the equirectangular (Plate Carrée) projection. Such a projection contains 360 degrees of horizontal view (a full wrap-around) and 180 degrees of vertical view (from straight up to straight down). These fields of view result in an image with an aspect ratio of 2:1. A full wrap-around panorama is shown below.

Panorama images are generally obtained by taking multiple photos from one position and stitching them together using panorama software. (See Wikipedia's [Comparison of photo stitching applications](#)

([https://en.wikipedia.org/wiki/Comparison_of_photo_stitching_applications_\(software\)](https://en.wikipedia.org/wiki/Comparison_of_photo_stitching_applications_(software))) for more information.) Such images should share a single "camera" locus, from which each of the panorama images are taken. The resulting 360 degree panorama can then define a projection on a sphere with the image wrapped to the two-dimensional surface of the sphere.



Treating the panorama as a projection on a sphere with a rectilinear coordinate system is advantageous when dividing up the image into rectilinear *tiles*, and serving images based on computed tile coordinates.

Creating Custom Panorama Tiles

Street View also supports different levels of image detail through the use of a zoom control, which allows you to zoom in and out from the default view. Generally, Street View provides five levels of zoom resolution for any given panorama image. If you were to rely on a single panorama image to serve all zoom levels, such an image would either necessarily be quite large and significantly slow down your application, or be of such poor resolution at higher zoom levels that you would serve a poorly pixellated image. Luckily, however, we can use a similar design pattern used to serve [Google's map tiles](#) (<https://developers.google.com/maps/documentation/javascript/coordinates#tile-coordinates>) at different zoom levels to provide appropriate resolution imagery for panoramas at each zoom level.

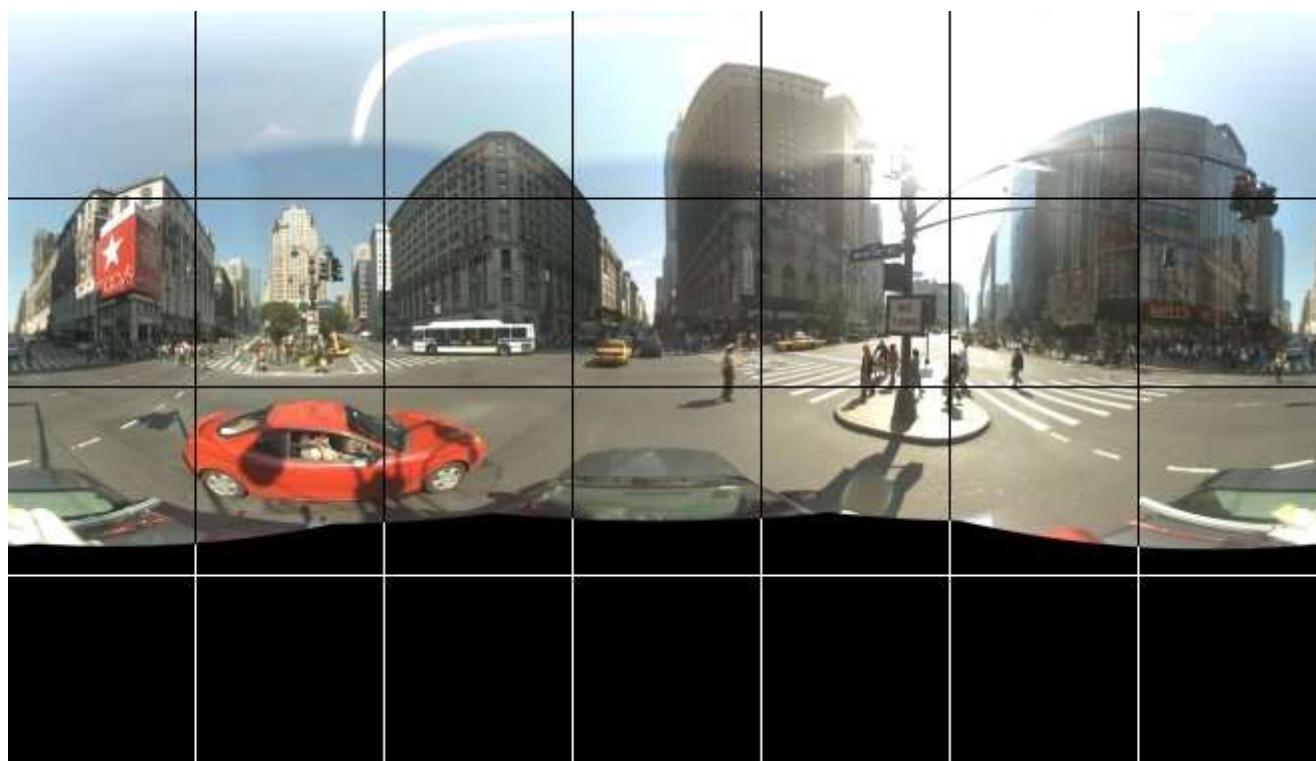
When a `StreetViewPanorama` first loads, by default it displays an image consisting of 25% (90 degrees of arc) of the horizontal breadth of the panorama at zoom level 1. This view corresponds roughly with a normal human field of view. Zooming "out" from this default view essentially provides a wider arc, while zooming in narrows the field of a view to a smaller arc. The `StreetViewPanorama` automatically calculates the appropriate field of view for the selected zoom level, and then selects imagery most appropriate for that resolution by selecting a tile set that roughly matches the dimensions of the horizontal field of view. The following fields of view map to Street View zoom levels:

Street View zoom level	Field of View (degrees)
0	180
1 (default)	90
2	45
3	22.5
4	11.25

Note that the size of the image shown within Street View is entirely dependent on the screen size (width) of the Street View container. If you provide a wider container, the service will still provide the same field of view for any given zoom level, though it may select tiles more appropriate for that resolution instead.

Because each panorama consists of an equirectangular projection, creating panorama tiles is relatively easy. As the projection provides an image with an aspect ratio of 2:1, tiles with 2:1 ratios are easier to use, though square tiles may provide better performance on square maps (since the field of view will be square).

For 2:1 tiles, a single image encompassing the entire panorama represents the entire panorama "world" (the base image) at zoom level 0, with each increasing zoom level offering $4^{\text{zoomLevel}}$ tiles. (E.g. at zoom level 2, the entire panorama consists of 16 tiles.) **Note:** zoom levels in Street View tiling do not match directly to zoom levels as provided using the Street View control; instead the Street View control zoom levels select a Field of View (FoV), from which appropriate tiles are selected.



Generally, you will want to name your image tiles so they can be selected programmatically. Such a naming scheme is discussed below in [Handling Custom Panorama Requests](#) (#CustomPanoramaMethod).

Handling Custom Panorama Requests

To use a custom panorama, call `StreetViewPanorama.registerPanoProvider()`, specifying the name of your custom panorama provider method. The panorama provider method must return a `StreetViewPanoramaData` object, and has the following signature:

```
Function(pano):StreetViewPanoramaData
```

A `StreetViewPanoramaData` is an object of the following form:

```
{
  copyright: string,
  location: {
    description: string,
    latLng: google.maps.LatLng,
    pano: string
  },
  tiles: {
    tileSize: google.maps.Size,
    worldSize: google.maps.Size,
    heading: number,
    getTileUrl: Function
  },
  links: [
    description: string,
    heading: number,
    pano: string,
    roadColor: string,
    roadOpacity: number
  ]
}
```

Display a custom panorama as follows:

- Set the `StreetViewPanoramaOptions.pano` (<https://developers.google.com/maps/documentation/javascript/3.exp/reference#StreetViewPanoramaOptions>) property to a custom value.
- Call `StreetViewPanorama.registerPanoProvider()` (<https://developers.google.com/maps/documentation/javascript/3.exp/reference#StreetViewPanorama>) to supply a custom panorama provider function.
- Implement your custom panorama provider function to handle the specified `pano` value.
- Construct a `StreetViewPanoramaData` (<https://developers.google.com/maps/documentation/javascript/3.exp/reference#StreetViewPanoramaData>) object.

- Set the `StreetViewTileData.getTileUrl` (<https://developers.google.com/maps/documentation/javascript/3.exp/reference#StreetViewTileData>) property to the name of a custom tile provider function that you supply. For example, `getCustomPanoramaTileUrl`.
- Implement your custom tile provider function, as shown in the samples below.
- Return the `StreetViewPanoramaData` object.

Note: do not directly set a `position` on the `StreetViewPanorama` when you wish to display custom panoramas, as such a position will instruct the Street View service to request the default Street View imagery close to that location. Instead, set this position within the custom `StreetViewPanoramaData` object's `location.latLng` field.

The following example displays a custom panorama of the Google Sydney office. Note that that this example doesn't use a map or default Street View imagery:

```
function initPano() {
  // Set up Street View and initially set it visible. Register the
  // custom panorama provider function. Set the StreetView to display
  // the custom panorama 'reception' which we check for below.
  var panorama = new google.maps.StreetViewPanorama(
    document.getElementById('map'), {pano: 'reception', visible: true});
  panorama.registerPanoProvider(getCustomPanorama);
}

// Return a pano image given the panoID.
function getCustomPanoramaTileUrl(pano, zoom, tileX, tileY) {
  return 'https://developers.google.com/maps/documentation/javascript/examples/full/images/'
    + 'panoReception1024-' + zoom + '-' + tileX + '-' + tileY + '.jpg';
}

// Construct the appropriate StreetViewPanoramaData given
// the passed pano IDs.
function getCustomPanorama(pano) {
  if (pano === 'reception') {
    return {
      location: {
        pano: 'reception',
        description: 'Google Sydney - Reception'
      },
      links: [],
      // The text for the copyright control.
      copyright: 'Imagery (c) 2010 Google',
      // The definition of the tiles for this panorama.
      tiles: {
        tileSize: new google.maps.Size(1024, 512),
        worldSize: new google.maps.Size(2048, 1024),
        // The heading in degrees at the origin of the panorama
        // tile set.
        centerHeading: 105,
        getTileUrl: getCustomPanoramaTileUrl
      }
    };
  }
}
```

[\[view example\]](#) (<https://developers.google.com/maps/documentation/javascript/examples/streetview-custom-simple>)

The custom panorama provider returns the appropriate tile given the passed panorama ID, zoom level, and panorama tile coordinates. Since image selection depends on these passed values, it is useful to name images that can be selected programmatically given those passed values, such as `pano_zoom_tileX_tileY.png`.

The following example adds another arrow to the image, in addition to the default Street View navigation arrows, that points into Google Sydney and links to the custom imagery:

```
var panorama;

// StreetViewPanoramaData of a panorama just outside the Google Sydney office.
var outsideGoogle;

// StreetViewPanoramaData for a custom panorama: the Google Sydney reception.
function getReceptionPanoramaData() {
```

```
return {
  location: {
    pano: 'reception', // The ID for this custom panorama.
    description: 'Google Sydney - Reception',
    latLng: new google.maps.LatLng(-33.86684, 151.19583)
  },
  links: [{
    heading: 195,
    description: 'Exit',
    pano: outsideGoogle.location.pano
  }],
  copyright: 'Imagery (c) 2010 Google',
  tiles: {
    tileSize: new google.maps.Size(1024, 512),
    worldSize: new google.maps.Size(2048, 1024),
    centerHeading: 105,
    getTileUrl: function(pano, zoom, tileX, tileY) {
      return 'https://developers.google.com/maps/documentation/javascript/examples/full/images/' +
        'panoReception1024-' + zoom + '-' + tileX + '-' + tileY + '.jpg';
    }
  }
};

function initPanorama() {
  panorama = new google.maps.StreetViewPanorama(
    document.getElementById('street-view'),
    {pano: outsideGoogle.location.pano});
  // Register a provider for the custom panorama.
  panorama.registerPanoProvider(function(pano) {
    if (pano === 'reception') {
      return getReceptionPanoramaData();
    }
    return null;
  });

  // Add a link to our custom panorama from outside the Google Sydney office.
  panorama.addListener('links_changed', function() {
    if (panorama.getPano() === outsideGoogle.location.pano) {
      panorama.getLinks().push({
        description: 'Google Sydney',
        heading: 25,
        pano: 'reception'
      });
    }
  });
}

function initialize() {
  // Use the Street View service to find a pano ID on Pirrama Rd, outside the
  // Google office.
  var streetviewService = new google.maps.StreetViewService;
  streetviewService.getPanorama(
    {location: {lat: -33.867386, lng: 151.195767}},
    function(result, status) {
      if (status === 'OK') {
        outsideGoogle = result;
        initPanorama();
      }
    });
}
```

[view example] (<https://developers.google.com/maps/documentation/javascript/examples/streetview-custom-tiles>)

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 3.0 License](http://creativecommons.org/licenses/by/3.0/) (<http://creativecommons.org/licenses/by/3.0/>), and code samples are licensed under the [Apache 2.0 License](http://www.apache.org/licenses/LICENSE-2.0) (<http://www.apache.org/licenses/LICENSE-2.0>). For details, see our [Site Policies](https://developers.google.com/terms/site-policies) (<https://developers.google.com/terms/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated August 3, 2017.

**Stack Overflow**

Ask a question under the
google-maps-api-3 tag.

**GitHub**

Fork our samples and try them
yourself.

**Blog**

Read the latest updates,
customer stories, and tips.

**Issue Tracker**

Something wrong? Send us a
bug report!