Google Maps JavaScript API V3 Reference

Experimental Version

Last updated Tuesday, September 01, 2015

Welcome to the reference for the experimental release of the Google JavaScript Maps API V3. This reference is kept up to date with the latest changes to the API. Please consult the [Javascript Maps API release notes] for information on what's new in each release.

Map class

`google.maps.Map` class

This class extends `MVCObject`.

Constructor	
`Map(mapDiv:Node, opts?:`*`MapOptions`*`)`	Creates a new map inside of the given HTML container, which is typically a `DIV` element.

Methods	
`fitBounds(bounds:LatLngBounds)`	**Return Value:** None Sets the viewport to contain the given bounds.
`getBounds()`	**Return Value:** `LatLngBounds` Returns the lat/lng bounds of the current viewport. If more than one copy of the world is visible, the bounds range in longitude from -180 to 180 degrees inclusive. If the map is not yet initialized (i.e. the mapType is still null), or center and zoom have not been set then the result is `null` or `undefined`.
`getCenter()`	**Return Value:** `LatLng` Returns the position displayed at the center of the map. Note that

	Returns the position displayed at the center of the map. Note that this `LatLng` object is *not* wrapped. See `LatLng` for more information.
`getDiv()`	**Return Value:** `Node`
`getHeading()`	**Return Value:** `number` Returns the compass heading of aerial imagery. The heading value is measured in degrees (clockwise) from cardinal direction North.
`getMapTypeId()`	**Return Value:** `MapTypeId|string`
`getProjection()`	**Return Value:** `Projection` Returns the current `Projection`. If the map is not yet initialized (i.e. the mapType is still null) then the result is null. Listen to `projection_changed` and check its value to ensure it is not null.
`getStreetView()`	**Return Value:** `StreetViewPanorama` Returns the default `StreetViewPanorama` bound to the map, which may be a default panorama embedded within the map, or the panorama set using `setStreetView()`. Changes to the map's `streetViewControl` will be reflected in the display of such a bound panorama.
`getTilt()`	**Return Value:** `number` Returns the current angle of incidence of the map, in degrees from the viewport plane to the map plane. The result will be `0` for imagery taken directly overhead or `45` for 45° imagery. 45° imagery is only available for `SATELLITE` and `HYBRID` map types, within some locations, and at some zoom levels. **Note:** This method does not return the value set by `setTilt`. See `setTilt` for details.
`getZoom()`	**Return Value:** `number`
`panBy(x:number, y:number)`	**Return Value:** None Changes the center of the map by the given distance in pixels. If the distance is less than both the width and height of the map, the transition will be smoothly animated. Note that the map coordinate system increases from west to east (for x values) and north to south (for y values).
`panTo(latLng:LatLng|LatLngLiteral)`	**Return Value:** None Changes the center of the map to the given `LatLng`. If the change is less than both the width and height of the map, the transition will

	be smoothly animated.
`panToBounds(latLngBounds:LatLngBounds)`	**Return Value:** None Pans the map by the minimum amount necessary to contain the given `LatLngBounds`. It makes no guarantee where on the map the bounds will be, except that as much of the bounds as possible will be visible. The bounds will be positioned inside the area bounded by the map type and navigation (pan, zoom, and Street View) controls, if they are present on the map. If the bounds is larger than the map, the map will be shifted to include the northwest corner of the bounds. If the change in the map's position is less than both the width and height of the map, the transition will be smoothly animated.
`setCenter(latlng:LatLng|LatLngLiteral)`	**Return Value:** None
`setHeading(heading:number)`	**Return Value:** None Sets the compass heading for aerial imagery measured in degrees from cardinal direction North.
`setMapTypeId(mapTypeId:MapTypeId|string)`	**Return Value:** None
`setOptions(options:`*`MapOptions`*`)`	**Return Value:** None
`setStreetView(panorama:StreetViewPanorama)`	**Return Value:** None Binds a `StreetViewPanorama` to the map. This panorama overrides the default `StreetViewPanorama`, allowing the map to bind to an external panorama outside of the map. Setting the panorama to `null` binds the default embedded panorama back to the map.
`setTilt(tilt:number)`	**Return Value:** None Controls the automatic switching behavior for the angle of incidence of the map. The only allowed values are `0` and 45. `setTilt(0)` causes the map to always use a 0° overhead view regardless of the zoom level and viewport. `setTilt(45)` causes the tilt angle to automatically switch to 45 whenever 45° imagery is available for the current zoom level and viewport, and switch back to 0 whenever 45° imagery is not available (this is the default behavior). 45° imagery is only available for `SATELLITE` and `HYBRID` map types, within some locations, and at some zoom levels. **Note:** `getTilt` returns the current tilt angle, not the value set by `setTilt`. Because `getTilt` and `setTilt` refer to different things, do not `bind()` the `tilt` property; doing so may yield unpredictable effects.

	effects.
`setZoom(zoom:number)`	**Return Value:** None

Properties	
`controls`	**Type:** `Array<MVCArray<Node>>` Additional controls to attach to the map. To add a control to the map, add the control's `<div>` to the `MVCArray` corresponding to the `ControlPosition` where it should be rendered.
`data`	**Type:** `Data` An instance of `Data`, bound to the map. Add features to this `Data` object to conveniently display them on this map.
`mapTypes`	**Type:** `MapTypeRegistry` A registry of `MapType` instances by string ID.
`overlayMapTypes`	**Type:** `MVCArray<MapType>` Additional map types to overlay.

Events	
`bounds_changed`	**Arguments:** None This event is fired when the viewport bounds have changed.
`center_changed`	**Arguments:** None This event is fired when the map center property changes.
`click`	**Arguments:** *`MouseEvent`* This event is fired when the user clicks on the map (but not when they click on a marker or infowindow).
`dblclick`	**Arguments:** *`MouseEvent`* This event is fired when the user double-clicks on the map. Note that the click event will also fire, right before this one.
`drag`	**Arguments:** None This event is repeatedly fired while the user drags the map.

`dragend`	**Arguments:** None This event is fired when the user stops dragging the map.
`dragstart`	**Arguments:** None This event is fired when the user starts dragging the map.
`heading_changed`	**Arguments:** None This event is fired when the map heading property changes.
`idle`	**Arguments:** None This event is fired when the map becomes idle after panning or zooming.
`maptypeid_changed`	**Arguments:** None This event is fired when the mapTypeId property changes.
`mousemove`	**Arguments:** *`MouseEvent`* This event is fired whenever the user's mouse moves over the map container.
`mouseout`	**Arguments:** *`MouseEvent`* This event is fired when the user's mouse exits the map container.
`mouseover`	**Arguments:** *`MouseEvent`* This event is fired when the user's mouse enters the map container.
`projection_changed`	**Arguments:** None This event is fired when the projection has changed.
`resize`	**Arguments:** None Developers should trigger this event on the map when the div changes size: `google.maps.event.trigger(map, 'resize')` .
`rightclick`	**Arguments:** *`MouseEvent`* This event is fired when the DOM contextmenu event is fired on the map container.
`tilesloaded`	**Arguments:** None This event is fired when the visible tiles have finished loading.
`tilt_changed`	**Arguments:** None This event is fired when the map tilt property changes.

`zoom_changed`	**Arguments:** None This event is fired when the map zoom property changes.

MapOptions object specification

`google.maps.MapOptions` object specification

Properties	
`backgroundColor`	**Type:** `string` Color used for the background of the Map div. This color will be visible when tiles have not yet loaded as the user pans. This option can only be set when the map is initialized.
`center`	**Type:** `LatLng` The initial Map center. Required.
`disableDefaultUI`	**Type:** `boolean` Enables/disables all default UI. May be overridden individually.
`disableDoubleClickZoom`	**Type:** `boolean` Enables/disables zoom and center on double click. Enabled by default.
`draggable`	**Type:** `boolean` If false, prevents the map from being dragged. Dragging is enabled by default.
`draggableCursor`	**Type:** `string` The name or url of the cursor to display when mousing over a draggable map. This property uses the css `cursor` attribute to change the icon. As with the css property, you must specify at least one fallback cursor that is not a URL. For example: `draggableCursor: 'url(http://www.example.com/icon.png), auto;'`.
`draggingCursor`	**Type:** `string` The name or url of the cursor to display when the map is being dragged. This property uses the css `cursor` attribute to change the icon. As with the css property, you must specify at least one fallback cursor that is not a URL. For example: `draggingCursor: 'url(http://www.example.com/icon.png), auto;'`.

heading	**Type:** number The heading for aerial imagery in degrees measured clockwise from cardinal direction North. Headings are snapped to the nearest available angle for which imagery is available.
keyboardShortcuts	**Type:** boolean If false, prevents the map from being controlled by the keyboard. Keyboard shortcuts are enabled by default.
mapMaker	**Type:** boolean True if Map Maker tiles should be used instead of regular tiles.
mapTypeControl	**Type:** boolean The initial enabled/disabled state of the Map type control.
mapTypeControlOptions	**Type:** MapTypeControlOptions The initial display options for the Map type control.
mapTypeId	**Type:** MapTypeId The initial Map mapTypeId. Defaults to ROADMAP.
maxZoom	**Type:** number The maximum zoom level which will be displayed on the map. If omitted, or set to null, the maximum zoom from the current map type is used instead.
minZoom	**Type:** number The minimum zoom level which will be displayed on the map. If omitted, or set to null, the minimum zoom from the current map type is used instead.
noClear	**Type:** boolean If true, do not clear the contents of the Map div.
overviewMapControl	**Type:** boolean The enabled/disabled state of the Overview Map control.
overviewMapControlOptions	**Type:** OverviewMapControlOptions The display options for the Overview Map control.
panControl	**Type:** boolean The enabled/disabled state of the Pan control.

panControlOptions	**Type:** PanControlOptions The display options for the Pan control.
rotateControl	**Type:** boolean The enabled/disabled state of the Rotate control.
rotateControlOptions	**Type:** RotateControlOptions The display options for the Rotate control.
scaleControl	**Type:** boolean The initial enabled/disabled state of the Scale control.
scaleControlOptions	**Type:** ScaleControlOptions The initial display options for the Scale control.
scrollwheel	**Type:** boolean If false, disables scrollwheel zooming on the map. The scrollwheel is enabled by default.
streetView	**Type:** StreetViewPanorama A StreetViewPanorama to display when the Street View pegman is dropped on the map. If no panorama is specified, a default StreetViewPanorama will be displayed in the map's div when the pegman is dropped.
streetViewControl	**Type:** boolean The initial enabled/disabled state of the Street View Pegman control. This control is part of the default UI, and should be set to false when displaying a map type on which the Street View road overlay should not appear (e.g. a non-Earth map type).
streetViewControlOptions	**Type:** StreetViewControlOptions The initial display options for the Street View Pegman control.
styles	**Type:** Array<MapTypeStyle> Styles to apply to each of the default map types. Note that for Satellite/Hybrid and Terrain modes, these styles will only apply to labels and geometry.
tilt	**Type:** number Controls the automatic switching behavior for the angle of incidence of the map. The only allowed values are 0 and 45. The value 0 causes the map to always use a 0° overhead view regardless of the zoom level and viewport. The value 45 causes the tilt angle to automatically switch to

	viewport. The value 45 causes the tilt angle to automatically switch to 45 whenever 45° imagery is available for the current zoom level and viewport, and switch back to 0 whenever 45° imagery is not available (this is the default behavior). 45° imagery is only available for `SATELLITE` and `HYBRID` map types, within some locations, and at some zoom levels. **Note:** `getTilt` returns the current tilt angle, not the value specified by this option. Because `getTilt` and this option refer to different things, do not `bind()` the `tilt` property; doing so may yield unpredictable effects.
`zoom`	**Type:** `number` The initial Map zoom level. Required.
`zoomControl`	**Type:** `boolean` The enabled/disabled state of the Zoom control.
`zoomControlOptions`	**Type:** `ZoomControlOptions` The display options for the Zoom control.

MapTypeId class

`google.maps.MapTypeId` class

Identifiers for common MapTypes.

Constant	
`HYBRID`	This map type displays a transparent layer of major streets on satellite images.
`ROADMAP`	This map type displays a normal street map.
`SATELLITE`	This map type displays satellite images.
`TERRAIN`	This map type displays maps with physical features such as terrain and vegetation.

MapTypeControlOptions object specification

`google.maps.MapTypeControlOptions` object specification

Options for the rendering of the map type control.

Properties	
mapTypeIds	**Type:** `Array<MapTypeId|string>` IDs of map types to show in the control.
position	**Type:** `ControlPosition` Position id. Used to specify the position of the control on the map. The default position is `TOP_RIGHT`.
style	**Type:** `MapTypeControlStyle` Style id. Used to select what style of map type control to display.

MapTypeControlStyle class

`google.maps.MapTypeControlStyle` class

Identifiers for common MapTypesControls.

Constant	
DEFAULT	Uses the default map type control. The control which DEFAULT maps to will vary according to window size and other factors. It may change in future versions of the API.
DROPDOWN_MENU	A dropdown menu for the screen realestate conscious.
HORIZONTAL_BAR	The standard horizontal radio buttons bar.

OverviewMapControlOptions object specification

`google.maps.OverviewMapControlOptions` object specification

Options for the rendering of the Overview Map control.

Properties	
opened	**Type:** `boolean` Whether the control should display in opened mode or collapsed (minimized) mode. By default, the control is closed.

PanControlOptions object specification

`google.maps.PanControlOptions` object specification

Options for the rendering of the pan control.

Properties	
`position`	**Type:** `ControlPosition` Position id. Used to specify the position of the control on the map. The default position is `TOP_LEFT`.

RotateControlOptions object specification

`google.maps.RotateControlOptions` object specification

Options for the rendering of the rotate control.

Properties	
`position`	**Type:** `ControlPosition` Position id. Used to specify the position of the control on the map. The default position is `TOP_LEFT`.

ScaleControlOptions object specification

`google.maps.ScaleControlOptions` object specification

Options for the rendering of the scale control.

Properties	
`style`	**Type:** `ScaleControlStyle` Style id. Used to select what style of scale control to display.

ScaleControlStyle class

`google.maps.ScaleControlStyle` class

Identifiers for scale control ids.

Constant	
`DEFAULT`	The standard scale control.

StreetViewControlOptions object specification

`google.maps.StreetViewControlOptions` object specification

Options for the rendering of the Street View pegman control on the map.

Properties	
`position`	**Type:** `ControlPosition` Position id. Used to specify the position of the control on the map. The default position is embedded within the navigation (zoom and pan) controls. If this position is empty or the same as that specified in the `zoomControlOptions` or `panControlOptions`, the Street View control will be displayed as part of the navigation controls. Otherwise, it will be displayed separately.

ZoomControlOptions object specification

`google.maps.ZoomControlOptions` object specification

Options for the rendering of the zoom control.

Properties	
`position`	**Type:** `ControlPosition` Position id. Used to specify the position of the control on the map. The default position is `TOP_LEFT`.

	TOP_LEFT.
style	**Type:** ZoomControlStyle Style id. Used to select what style of zoom control to display.

ZoomControlStyle class

`google.maps.ZoomControlStyle` class

Identifiers for the zoom control.

Constant	
DEFAULT	The default zoom control. The control which DEFAULT maps to will vary according to map size and other factors. It may change in future versions of the API.
LARGE	The larger control, with the zoom slider in addition to +/- buttons.
SMALL	A small control with buttons to zoom in and out.

ControlPosition class

`google.maps.ControlPosition` class

Identifiers used to specify the placement of controls on the map. Controls are positioned relative to other controls in the same layout position. Controls that are added first are positioned closer to the edge of the map.

```
  +----------------+
  + TL    TC    TR +
  + LT          RT +
  +                +
  + LC          RC +
  +                +
  + LB          RB +
  + BL    BC    BR +
  +----------------+
```

Elements in the top or bottom row flow towards the middle of the row. Elements in the left or right column flow towards the middle of the column.

Constant	
`BOTTOM_CENTER`	Elements are positioned in the center of the bottom row.
`BOTTOM_LEFT`	Elements are positioned in the bottom left and flow towards the middle. Elements are positioned to the right of the Google logo.
`BOTTOM_RIGHT`	Elements are positioned in the bottom right and flow towards the middle. Elements are positioned to the left of the copyrights.
`LEFT_BOTTOM`	Elements are positioned on the left, above bottom-left elements, and flow upwards.
`LEFT_CENTER`	Elements are positioned in the center of the left side.
`LEFT_TOP`	Elements are positioned on the left, below top-left elements, and flow downwards.
`RIGHT_BOTTOM`	Elements are positioned on the right, above bottom-right elements, and flow upwards.
`RIGHT_CENTER`	Elements are positioned in the center of the right side.
`RIGHT_TOP`	Elements are positioned on the right, below top-right elements, and flow downwards.
`TOP_CENTER`	Elements are positioned in the center of the top row.
`TOP_LEFT`	Elements are positioned in the top left and flow towards the middle.
`TOP_RIGHT`	Elements are positioned in the top right and flow towards the middle.

Data class

`google.maps.Data` class

A layer for displaying geospatial data. Points, line-strings and polygons can be displayed.

Every `Map` has a `Data` object by default, so most of the time there is no need to construct one. For example:

```
var myMap = new google.maps.Map(...);
myMap.data.addGeoJson(...);
myMap.data.setStyle(...);
```

The `Data` object is a collection of `Features`.

This class extends `MVCObject`.

Constructor	
`Data(options?:Data.DataOptions)`	Creates an empty collection, with the given `DataOptions`.

Methods	
`add(feature:Data.Feature|Data.FeatureOptions)`	**Return Value:** `Data.Feature` Adds a feature to the collection, and returns the added feature. If the feature has an ID, it will replace any existing feature in the collection with the same ID. If no feature is given, a new feature will be created with null geometry and no properties. If `FeatureOptions` are given, a new feature will be created with the specified properties. Note that the IDs 1234 and `'1234'` are equivalent. Adding a feature with ID 1234 will replace a feature with ID `'1234'`, and vice versa.
`addGeoJson(geoJson:Object, options?:Data.GeoJsonOptions)`	**Return Value:** `Array<Data.Feature>` Adds GeoJSON features to the collection. Give this method a parsed JSON. The imported features are returned. Throws an exception if the GeoJSON could not be imported.
`contains(feature:Data.Feature)`	**Return Value:** `boolean` Checks whether the given feature is in the collection.
`forEach(callback:function(Data.Feature))`	**Return Value:** None Repeatedly invokes the given function, passing a feature in the collection to the function on each invocation. The order of iteration through the features is undefined.
`getControlPosition()`	**Return Value:** `ControlPosition` Returns the position of the drawing controls on the map.
`getControls()`	**Return Value:** `Array<string>` Returns which drawing modes are available for the user to select, in the order they are displayed. This does not include the null drawing mode, which is added by default. Possible drawing modes are `"Point"`, `"LineString"` or `"Polygon"`.
`getDrawingMode()`	**Return Value:** `string` Returns the current drawing mode of the given Data layer. A

	drawing mode of null means that the user can interact with the map as normal, and clicks do not draw anything. Possible drawing modes are `null`, `"Point"`, `"LineString"` or `"Polygon"`.
`getFeatureById(id:number|string)`	**Return Value:** `Data.Feature|undefined` Returns the feature with the given ID, if it exists in the collection. Otherwise returns `undefined`. Note that the IDs `1234` and `'1234'` are equivalent. Either can be used to look up the same feature.
`getMap()`	**Return Value:** `Map` Returns the map on which the features are displayed.
`getStyle()`	**Return Value:** `Data.StylingFunction|Data.StyleOptions` Gets the style for all features in the collection.
`loadGeoJson(url:string, options?:Data.GeoJsonOptions, callback?:function(Array<Data.Feature>))`	**Return Value:** None Loads GeoJSON from a URL, and adds the features to the collection.
`overrideStyle(feature:Data.Feature, style:Data.StyleOptions)`	**Return Value:** None Changes the style of a feature. These changes are applied on top of the style specified by `setStyle()`. Style properties set to null revert to the value specified via `setStyle()`.
`remove(feature:Data.Feature)`	**Return Value:** None Removes a feature from the collection.
`revertStyle(feature?:Data.Feature)`	**Return Value:** None Removes the effect of previous `overrideStyle()` calls. The style of the given feature reverts to the style specified by `setStyle()`. If no feature is given, all features have their style reverted.
`setControlPosition(controlPosition:ControlPosition)`	**Return Value:** None Sets the position of the drawing controls on the map.
`setControls(controls:Array<string>)`	**Return Value:** None

`controls:Array<string>)`	Sets which drawing modes are available for the user to select, in the order they are displayed. This should not include the null drawing mode, which is added by default. If null, drawing controls are disabled and not displayed. Possible drawing modes are `"Point"`, `"LineString"` or `"Polygon"`.
`setDrawingMode(drawingMode:string)`	**Return Value:** None Sets the current drawing mode of the given Data layer. A drawing mode of null means that the user can interact with the map as normal, and clicks do not draw anything. Possible drawing modes are `null`, `"Point"`, `"LineString"` or `"Polygon"`.
`setMap(map:Map)`	**Return Value:** None Renders the features on the specified map. If map is set to null, the features will be removed from the map.
`setStyle(style:Data.StylingFunction|Data.StyleOptions)`	**Return Value:** None Sets the style for all features in the collection. Styles specified on a per-feature basis via `overrideStyle()` continue to apply. Pass either an object with the desired style options, or a function that computes the style for each feature. The function will be called every time a feature's properties are updated.
`toGeoJson(callback:function(Object))`	**Return Value:** None Exports the features in the collection to a GeoJSON object.

Events	
`addfeature`	**Arguments:** *`Data.AddFeatureEvent`* This event is fired when a feature is added to the collection.
`click`	**Arguments:** *`Data.MouseEvent`* This event is fired for a click on the geometry.
`dblclick`	**Arguments:** *`Data.MouseEvent`* This event is fired for a double click on the geometry.
`mousedown`	**Arguments:** *`Data.MouseEvent`* This event is fired for a mousedown on the geometry.

`mouseout`	**Arguments:** *Data.MouseEvent* This event is fired when the mouse leaves the area of the geometry.
`mouseover`	**Arguments:** *Data.MouseEvent* This event is fired when the mouse enters the area of the geometry.
`mouseup`	**Arguments:** *Data.MouseEvent* This event is fired for a mouseup on the geometry.
`removefeature`	**Arguments:** *Data.RemoveFeatureEvent* This event is fired when a feature is removed from the collection.
`removeproperty`	**Arguments:** *Data.RemovePropertyEvent* This event is fired when a feature's property is removed.
`rightclick`	**Arguments:** *Data.MouseEvent* This event is fired for a rightclick on the geometry.
`setgeometry`	**Arguments:** *Data.SetGeometryEvent* This event is fired when a feature's geometry is set.
`setproperty`	**Arguments:** *Data.SetPropertyEvent* This event is fired when a feature's property is set.

Data.DataOptions object specification

`google.maps.Data.DataOptions` object specification

DataOptions object used to define the properties that a developer can set on a `Data` object.

Properties	
`controlPosition`	**Type:** `ControlPosition` The position of the drawing controls on the map. The default position is TOP_LEFT.
`controls`	**Type:** `Array<string>` Describes which drawing modes are available for the user to select, in the order they are displayed. This should not include the null drawing mode, which is added by default. If null, drawing controls are disabled and not displayed. Defaults to null.

	default. If null, drawing controls are disabled and not displayed. Defaults to null. Possible drawing modes are `"Point"`, `"LineString"` or `"Polygon"`.
`drawingMode`	**Type:** `string` The current drawing mode of the given Data layer. A drawing mode of null means that the user can interact with the map as normal, and clicks do not draw anything. Defaults to null. Possible drawing modes are null, `"Point"`, `"LineString"` or `"Polygon"`.
`featureFactory`	**Type:** `function(Data.Geometry): Data.Feature` When drawing is enabled and a user draws a Geometry (a Point, Line String or Polygon), this function is called with that Geometry and should return a Feature that is to be added to the Data layer. If a featureFactory is not supplied, a Feature with no id and no properties will be created from that Geometry instead. Defaults to null.
`map`	**Type:** `Map` Map on which to display the features in the collection.
`style`	**Type:** `Data.StylingFunction|Data.StyleOptions` Style for all features in the collection. For more details, see the `setStyle()` method above.

Data.GeoJsonOptions object specification

`google.maps.Data.GeoJsonOptions` object specification

Optional parameters for importing GeoJSON.

Properties	
`idPropertyName`	**Type:** `string` The name of the Feature property to use as the feature ID. If not specified, the GeoJSON Feature id will be used.

Data.StyleOptions object specification

`google.maps.Data.StyleOptions` object specification

These options specify the way a Feature should appear when displayed on a map.

Properties	
`clickable`	**Type:** `boolean` If `true`, the marker receives mouse and touch events. Default value is `true`.
`cursor`	**Type:** `string` Mouse cursor to show on hover. Only applies to point geometries.
`draggable`	**Type:** `boolean` If `true`, the object can be dragged across the map and the underlying feature will have its geometry updated. Default value is `false`.
`editable`	**Type:** `boolean` If `true`, the object can be edited by dragging control points and the underlying feature will have its geometry updated. Only applies to LineString and Polygon geometries. Default value is `false`.
`fillColor`	**Type:** `string` The fill color. All CSS3 colors are supported except for extended named colors. Only applies to polygon geometries.
`fillOpacity`	**Type:** `number` The fill opacity between 0.0 and 1.0. Only applies to polygon geometries.
`icon`	**Type:** `string`|`Icon`|`Symbol` Icon for the foreground. If a string is provided, it is treated as though it were an `Icon` with the string as `url`. Only applies to point geometries.
`shape`	**Type:** `MarkerShape` Defines the image map used for hit detection. Only applies to point geometries.
`strokeColor`	**Type:** `string` The stroke color. All CSS3 colors are supported except for extended named colors. Only applies to line and polygon geometries.
`strokeOpacity`	**Type:** `number` The stroke opacity between 0.0 and 1.0. Only applies to line and polygon geometries.
`strokeWeight`	**Type:** `number` The stroke width in pixels. Only applies to line and polygon geometries.
`title`	**Type:** `string` Rollover text. Only applies to point geometries.

	Rollover text. Only applies to point geometries.
`visible`	**Type:** `boolean` Whether the feature is visible. Defaults to `true`.
`zIndex`	**Type:** `number` All features are displayed on the map in order of their zIndex, with higher values displaying in front of features with lower values. Markers are always displayed in front of line-strings and polygons.

Data.StylingFunction typedef

`google.maps.Data.StylingFunction` typedef

A function that computes the appearance of a feature.

The `Data.setStyle()` method can accept a styling function. Use this when features should appear differently depending on their properties. You can find more information about styling features in the developer's guide.

`function(Data.Feature):Data.StyleOptions`

Data.Feature class

`google.maps.Data.Feature` class

A feature has a geometry, an id, and a set of properties.

Constructor	
`Data.Feature(options?:Data.FeatureOptions)`	Constructs a Feature with the given options.

Methods	
`forEachProperty(callback:function(*, string))`	**Return Value:** None Repeatedly invokes the given function, passing a property value and name on each invocation. The order of iteration through the properties is undefined.

getGeometry()	**Return Value:** Data.Geometry Returns the feature's geometry.
getId()	**Return Value:** number|string|undefined Returns the feature ID.
getProperty(name:string)	**Return Value:** * Returns the value of the requested property, or undefined if the property does not exist.
removeProperty(name:string)	**Return Value:** None Removes the property with the given name.
setGeometry(newGeometry:Data.Geometry|LatLng|LatLngLiteral)	**Return Value:** None Sets the feature's geometry.
setProperty(name:string, newValue:*)	**Return Value:** None Sets the value of the specified property. If newValue is undefined this is equivalent to calling removeProperty.
toGeoJson(callback:function(Object))	**Return Value:** None Exports the feature to a GeoJSON object.

Events	
removeproperty	**Arguments:** Data.RemovePropertyEvent This event is triggered when a feature's property is removed.
setgeometry	**Arguments:** Data.SetGeometryEvent This event is triggered when a feature's geometry is set.
setproperty	**Arguments:** Data.SetPropertyEvent This event is triggered when a feature's property is set.

Data.FeatureOptions object specification

`google.maps.Data.FeatureOptions` object specification

Optional parameters for creating `Data.Feature` objects.

Properties	
`geometry`	**Type:** `Data.Geometry|LatLng|LatLngLiteral` The feature geometry. If none is specified when a feature is constructed, the feature's geometry will be null. If a `LatLng` object or `LatLngLiteral` is given, this will be converted to a `Data.Point` geometry.
`id`	**Type:** `number|string` Feature ID is optional. If provided, it can be used to look up the feature in a `Data` object using the `getFeatureById()` method. Note that a feature's ID cannot be subsequently changed.
`properties`	**Type:** `Object` The feature properties. This is an arbitrary mapping of property names to values.

Data.Geometry class

`google.maps.Data.Geometry` class

A superclass for the various geometry objects.

Methods	
`getType()`	**Return Value:** `string` Returns the type of the geometry object. Possibilities are `"Point"`, `"MultiPoint"`, `"LineString"`, `"MultiLineString"`, `"LinearRing"`, `"Polygon"`, `"MultiPolygon"`, or `"GeometryCollection"`.

Data.Point class

`google.maps.Data.Point` class

A Point geometry contains a single `LatLng`.

This class extends `Data.Geometry`.

Constructor	
`Data.Point(latLng:LatLng|LatLngLiteral)`	Constructs a `Data.Point` from the given `LatLng` or `LatLngLiteral`.

Methods	
`get()`	**Return Value:** `LatLng` Returns the contained `LatLng`.
`getType()`	**Return Value:** `string` Returns the string `"Point"`.

Data.MultiPoint class

`google.maps.Data.MultiPoint` class

A MultiPoint geometry contains a number of `LatLngs`.

This class extends `Data.Geometry`.

Constructor	
`Data.MultiPoint(elements:Array<LatLng|LatLngLiteral>)`	Constructs a `Data.MultiPoint` from the given `LatLngs` or `LatLngLiterals`.

Methods	
`getArray()`	**Return Value:** `Array<LatLng>` Returns an array of the contained `LatLngs`. A new array is returned each time `getArray()` is called.
`getAt(n:number)`	**Return Value:** `LatLng` Returns the n-th contained `LatLng`.
`getLength()`	**Return Value:** `number`

	Returns the number of contained LatLngs.
getType()	**Return Value:** string Returns the string "MultiPoint".

Data.LineString class

google.maps.Data.LineString class

A LineString geometry contains a number of LatLngs.

This class extends Data.Geometry.

Constructor	
Data.LineString(elements:Array<LatLng|LatLngLiteral>)	Constructs a Data.LineString from the given LatLngs or LatLngLiterals.

Methods	
getArray()	**Return Value:** Array<LatLng> Returns an array of the contained LatLngs. A new array is returned each time getArray() is called.
getAt(n:number)	**Return Value:** LatLng Returns the n-th contained LatLng.
getLength()	**Return Value:** number Returns the number of contained LatLngs.
getType()	**Return Value:** string Returns the string "LineString".

Data.MultiLineString class

google.maps.Data.MultiLineString class

A MultiLineString geometry contains a number of `LineString`s.

This class extends `Data.Geometry`.

Constructor	
`Data.MultiLineString(elements:Array<Data.LineString|Array<LatLng|LatLngLiteral>>)`	Constructs a `Data.MultiLineString` from the given `Data.LineString`s or arrays of positions.

Methods	
`getArray()`	**Return Value:** `Array<Data.LineString>` Returns an array of the contained `Data.LineString`s. A new array is returned each time `getArray()` is called.
`getAt(n:number)`	**Return Value:** `Data.LineString` Returns the n-th contained `Data.LineString`.
`getLength()`	**Return Value:** `number` Returns the number of contained `Data.LineString`s.
`getType()`	**Return Value:** `string` Returns the string `"MultiLineString"`.

Data.LinearRing class

`google.maps.Data.LinearRing` class

A LinearRing geometry contains a number of `LatLng`s, representing a closed LineString. There is no need to make the first `LatLng` equal to the last `LatLng`. The LinearRing is closed implicitly.

This class extends `Data.Geometry`.

Constructor	
`Data.LinearRing(elements:Array<LatLng|LatLngLiteral>)`	Constructs a `Data.LinearRing` from the given `LatLng`s or `LatLngLiteral`s.

Methods	
`getArray()`	**Return Value:** `Array<LatLng>` Returns an array of the contained `LatLng`s. A new array is returned each time `getArray()` is called.
`getAt(n:number)`	**Return Value:** `LatLng` Returns the n-th contained `LatLng`.
`getLength()`	**Return Value:** `number` Returns the number of contained `LatLng`s.
`getType()`	**Return Value:** `string` Returns the string `"LinearRing"`.

Data.Polygon class

`google.maps.Data.Polygon` class

A Polygon geometry contains a number of `Data.LinearRing`s. The first linear-ring must be the polygon exterior boundary, and subsequent linear-rings must be interior boundaries or "holes".

This class extends `Data.Geometry`.

Constructor	
`Data.Polygon(elements:Array<Data.LinearRing|Array<LatLng|LatLngLiteral>>)`	Constructs a `Data.Polygon` from the given `Data.LinearRing`s or arrays of positions.

Methods	
`getArray()`	**Return Value:** `Array<Data.LinearRing>` Returns an array of the contained `Data.LinearRing`s. A new array is returned each time `getArray()` is called.
`getAt(n:number)`	**Return Value:** `Data.LinearRing` Returns the n-th contained `Data.LinearRing`.

getLength()	**Return Value:** number Returns the number of contained Data.LinearRings.
getType()	**Return Value:** string Returns the string "Polygon".

Data.MultiPolygon class

google.maps.Data.MultiPolygon class

A MultiPolygon geometry contains a number of Data.Polygons.

This class extends Data.Geometry.

Constructor	
Data.MultiPolygon(elements:Array<Data.Polygon|Array<Data.LinearRing|Array<LatLng|LatLngLiteral>>>)	Constructs a Data.MultiPolygon from the given Data.Polygons or arrays of positions.

Methods	
getArray()	**Return Value:** Array<Data.Polygon> Returns an array of the contained Data.Polygons. A new array is returned each time getArray() is called.
getAt(n:number)	**Return Value:** Data.Polygon Returns the n-th contained Data.Polygon.
getLength()	**Return Value:** number Returns the number of contained Data.Polygons.
getType()	**Return Value:** string Returns the string "MultiPolygon".

Data.GeometryCollection class

`google.maps.Data.GeometryCollection` class

A GeometryCollection contains a number of geometry objects. Any `LatLng` or `LatLngLiteral` objects are automatically converted to `Data.Point` geometry objects.

This class extends `Data.Geometry`.

Constructor	
`Data.GeometryCollection(elements:Array<Data.Geometry|LatLng|LatLngLiteral>)`	Constructs a `Data.GeometryCollection` from the given geometry objects or `LatLng`s.

Methods	
`getArray()`	**Return Value:** `Array<Data.Geometry>` Returns an array of the contained geometry objects. A new array is returned each time `getArray()` is called.
`getAt(n:number)`	**Return Value:** `Data.Geometry` Returns the n-th contained geometry object.
`getLength()`	**Return Value:** `number` Returns the number of contained geometry objects.
`getType()`	**Return Value:** `string` Returns the string `"GeometryCollection"`.

Data.MouseEvent object specification

`google.maps.Data.MouseEvent` object specification

This object is passed to mouse event handlers on a `Data` object.

This object extends `MouseEvent`.

Properties

feature	**Type:** Data.Feature The feature which generated the mouse event.

Data.AddFeatureEvent object specification

`google.maps.Data.AddFeatureEvent` object specification

The properties of a `addfeature` event.

Properties	
feature	**Type:** Data.Feature The feature that was added to the `FeatureCollection`.

Data.RemoveFeatureEvent object specification

`google.maps.Data.RemoveFeatureEvent` object specification

The properties of a `removefeature` event.

Properties	
feature	**Type:** Data.Feature The feature that was removed from the `FeatureCollection`.

Data.SetGeometryEvent object specification

`google.maps.Data.SetGeometryEvent` object specification

The properties of a `setgeometry` event.

Properties	
feature	**Type:** Data.Feature

feature	**Type:** Data.Feature The feature whose geometry was set.
newGeometry	**Type:** Data.Geometry The new feature geometry.
oldGeometry	**Type:** Data.Geometry The previous feature geometry.

Data.SetPropertyEvent object specification

`google.maps.Data.SetPropertyEvent` object specification

The properties of a `setproperty` event.

Properties	
feature	**Type:** Data.Feature The feature whose property was set.
name	**Type:** `string` The property name.
newValue	**Type:** `*` The new value.
oldValue	**Type:** `*` The previous value. Will be `undefined` if the property was added.

Data.RemovePropertyEvent object specification

`google.maps.Data.RemovePropertyEvent` object specification

The properties of a `removeproperty` event.

Properties	
feature	**Type:** Data.Feature

	The feature whose property was removed.
name	**Type:** `string` The property name.
oldValue	**Type:** `*` The previous value.

Marker class

`google.maps.Marker` class

This class extends `MVCObject`.

Constructor	
`Marker(opts?:MarkerOptions)`	Creates a marker with the options specified. If a map is specified, the marker is added to the map upon construction. Note that the position must be set for the marker to display.

Methods	
`getAnimation()`	**Return Value:** `Animation`
`getAttribution()`	**Return Value:** `Attribution`
`getClickable()`	**Return Value:** `boolean`
`getCursor()`	**Return Value:** `string`
`getDraggable()`	**Return Value:** `boolean`
`getIcon()`	**Return Value:** `string|Icon|Symbol`
`getLabel()`	**Return Value:** `MarkerLabel`
`getMap()`	**Return Value:** `Map|StreetViewPanorama`

getOpacity()	**Return Value:** number
getPlace()	**Return Value:** MarkerPlace
getPosition()	**Return Value:** LatLng
getShape()	**Return Value:** MarkerShape
getTitle()	**Return Value:** string
getVisible()	**Return Value:** boolean
getZIndex()	**Return Value:** number
setAnimation(animation:Animation)	**Return Value:** None Start an animation. Any ongoing animation will be cancelled. Currently supported animations are: BOUNCE, DROP. Passing in null will cause any animation to stop.
setAttribution(attribution:*Attribution*)	**Return Value:** None
setClickable(flag:boolean)	**Return Value:** None
setCursor(cursor:string)	**Return Value:** None
setDraggable(flag:boolean)	**Return Value:** None
setIcon(icon:string|Icon|Symbol)	**Return Value:** None
setLabel(label:string|MarkerLabel)	**Return Value:** None
setMap(map:Map|StreetViewPanorama)	**Return Value:** None Renders the marker on the specified map or panorama. If map is set to null, the marker will be removed.
setOpacity(opacity:number)	**Return Value:** None
setOptions(options:*MarkerOptions*)	**Return Value:** None

`options:`*`MarkerOptions`*`)`	

`setPlace(`
`place:`*`MarkerPlace`*`)`
	Return Value: None

`setPosition(`
`latlng:LatLng|`
`LatLngLiteral)`
	Return Value: None

`setShape(`
`shape:`*`MarkerShape`*`)`
	Return Value: None

`setTitle(title:string)`	**Return Value:** None

`setVisible(`
`visible:boolean)`
	Return Value: None

`setZIndex(zIndex:number)`	**Return Value:** None

Constant

`MAX_ZINDEX`	The maximum default z-index that the API will assign to a marker. You may set a higher z-index to bring a marker to the front.

Events

`animation_changed`	**Arguments:** None This event is fired when the marker's animation property changes.
`click`	**Arguments:** *`MouseEvent`* This event is fired when the marker icon was clicked.
`clickable_changed`	**Arguments:** None This event is fired when the marker's clickable property changes.
`cursor_changed`	**Arguments:** None This event is fired when the marker's cursor property changes.
`dblclick`	**Arguments:** *`MouseEvent`* This event is fired when the marker icon was double clicked.
`drag`	**Arguments:** *`MouseEvent`* This event is repeatedly fired while the user drags the marker.

	This event is repeatedly fired while the user drags the marker.
`dragend`	**Arguments:** *MouseEvent* This event is fired when the user stops dragging the marker.
`draggable_changed`	**Arguments:** None This event is fired when the marker's draggable property changes.
`dragstart`	**Arguments:** *MouseEvent* This event is fired when the user starts dragging the marker.
`flat_changed`	**Arguments:** None This event is fired when the marker's flat property changes.
`icon_changed`	**Arguments:** None This event is fired when the marker icon property changes.
`mousedown`	**Arguments:** *MouseEvent* This event is fired for a mousedown on the marker.
`mouseout`	**Arguments:** *MouseEvent* This event is fired when the mouse leaves the area of the marker icon.
`mouseover`	**Arguments:** *MouseEvent* This event is fired when the mouse enters the area of the marker icon.
`mouseup`	**Arguments:** *MouseEvent* This event is fired for a mouseup on the marker.
`position_changed`	**Arguments:** None This event is fired when the marker position property changes.
`rightclick`	**Arguments:** *MouseEvent* This event is fired for a rightclick on the marker.
`shape_changed`	**Arguments:** None This event is fired when the marker's shape property changes.
`title_changed`	**Arguments:** None This event is fired when the marker title property changes.
`visible_changed`	**Arguments:** None

	This event is fired when the marker's visible property changes.
`zindex_changed`	**Arguments:** None This event is fired when the marker's zIndex property changes.

MarkerOptions object specification

`google.maps.MarkerOptions` object specification

Properties	
`anchorPoint`	**Type:** `Point` The offset from the marker's position to the tip of an InfoWindow that has been opened with the marker as anchor.
`animation`	**Type:** `Animation` Which animation to play when marker is added to a map.
`attribution`	**Type:** `Attribution` Contains all the information needed to identify your application as the source of a save. In this context, 'place' means a business, point of interest or geographic location. `attribution` must be specified with a `place` in order to enable a save.
`clickable`	**Type:** `boolean` If true, the marker receives mouse and touch events. Default value is true.
`crossOnDrag`	**Type:** `boolean` If false, disables cross that appears beneath the marker when dragging. This option is `true` by default.
`cursor`	**Type:** `string` Mouse cursor to show on hover
`draggable`	**Type:** `boolean` If true, the marker can be dragged. Default value is false.
`icon`	**Type:** `string|Icon|Symbol` Icon for the foreground. If a string is provided, it is treated as though it were an `Icon` with the string as `url`.

`label`	**Type:** `string|MarkerLabel` Adds a label to the marker. The label can either be a string, or a `MarkerLabel` object. Only the first character of the string will be displayed.
`map`	**Type:** `Map|StreetViewPanorama` Map on which to display Marker.
`opacity`	**Type:** `number` The marker's opacity between 0.0 and 1.0.
`optimized`	**Type:** `boolean` Optimization renders many markers as a single static element. Optimized rendering is enabled by default. Disable optimized rendering for animated GIFs or PNGs, or when each marker must be rendered as a separate DOM element (advanced usage only).
`place`	**Type:** `MarkerPlace` Place information, used to identify and describe the place associated with this `Marker`. In this context, 'place' means a business, point of interest or geographic location. To allow a user to save this place, open an info window anchored on this marker. The info window will contain information about the place and an option for the user to save it. Only one of position or place can be specified.
`position`	**Type:** `LatLng` Marker position. Required.
`shape`	**Type:** `MarkerShape` Image map region definition used for drag/click.
`title`	**Type:** `string` Rollover text
`visible`	**Type:** `boolean` If true, the marker is visible
`zIndex`	**Type:** `number` All markers are displayed on the map in order of their zIndex, with higher values displaying in front of markers with lower values. By default, markers are displayed according to their vertical position on screen, with lower markers appearing in front of markers further up the screen.

Icon object specification

`google.maps.Icon` object specification

Properties	
`anchor`	**Type:** Point The position at which to anchor an image in correspondence to the location of the marker on the map. By default, the anchor is located along the center point of the bottom of the image.
`labelOrigin`	**Type:** Point The origin of the label relative to the top-left corner of the icon image, if a label is supplied by the marker. By default, the origin is located in the center point of the image.
`origin`	**Type:** Point The position of the image within a sprite, if any. By default, the origin is located at the top left corner of the image `(0, 0)`.
`scaledSize`	**Type:** Size The size of the entire image after scaling, if any. Use this property to stretch/shrink an image or a sprite.
`size`	**Type:** Size The display size of the sprite or image. When using sprites, you must specify the sprite size. If the size is not provided, it will be set when the image loads.
`url`	**Type:** `string` The URL of the image or sprite sheet.

MarkerLabel object specification

`google.maps.MarkerLabel` object specification

These options specify the appearance of a marker label. A marker label is a single character of text which will appear inside the marker. If you are using it with a custom marker, you can reposition it with the `labelOrigin` property in the `Icon` class.

Properties	

`color`	**Type:** `string` The color of the label text. Default color is black.
`fontFamily`	**Type:** `string` The font family of the label text (equivalent to the CSS font-family property).
`fontSize`	**Type:** `string` The font size of the label text (equivalent to the CSS font-size property). Default size is 14px.
`fontWeight`	**Type:** `string` The font weight of the label text (equivalent to the CSS font-weight property).
`text`	**Type:** `string` The text to be displayed in the label. Only the first character of this string will be shown.

MarkerShape object specification

`google.maps.MarkerShape` object specification

This object defines the clickable region of a marker image for browsers other than Internet Explorer. The shape consists of two properties — `type` and `coord` — which define the non-transparent region of an image. A `MarkerShape` object is not required on Internet Explorer since the browser does not fire events on the transparent region of an image by default.

Properties	
`coords`	**Type:** `Array<number>` The format of this attribute depends on the value of the `type` and follows the w3 AREA `coords` specification found at http://www.w3.org/TR/REC-html40/struct/objects.html#adef-coords. The `coords` attribute is an array of integers that specify the pixel position of the shape relative to the top-left corner of the target image. The coordinates depend on the value of `type` as follows: - `circle`: coords is `[x1,y1,r]` where x1,y2 are the coordinates of the center of the circle, and r is the radius of the circle. - `poly`: coords is `[x1,y1,x2,y2...xn,yn]` where each x,y pair contains the coordinates of one vertex of the polygon.

	- `rect`: coords is `[x1,y1,x2,y2]` where x1,y1 are the coordinates of the upper-left corner of the rectangle and x2,y2 are the coordinates of the lower-right coordinates of the rectangle.
`type`	**Type:** `string` Describes the shape's type and can be `circle`, `poly` or `rect`.

Symbol object specification

`google.maps.Symbol` object specification

Properties	
`anchor`	**Type:** `Point` The position of the symbol relative to the marker or polyline. The coordinates of the symbol's path are translated left and up by the anchor's x and y coordinates respectively. By default, a symbol is anchored at `(0, 0)`. The position is expressed in the same coordinate system as the symbol's path.
`fillColor`	**Type:** `string` The symbol's fill color. All CSS3 colors are supported except for extended named colors. For symbol markers, this defaults to 'black'. For symbols on polylines, this defaults to the stroke color of the corresponding polyline.
`fillOpacity`	**Type:** `number` The symbol's fill opacity. Defaults to 0.
`labelOrigin`	**Type:** `Point` The origin of the label relative to the origin of the path, if label is supplied by the marker. By default, the origin is located at `(0, 0)`. The origin is expressed in the same coordinate system as the symbol's path. This property is unused for symbols on polylines.
`path`	**Type:** `SymbolPath|string` The symbol's path, which is a built-in symbol path, or a custom path expressed using SVG path notation. Required.
`rotation`	**Type:** `number` The angle by which to rotate the symbol, expressed clockwise in degrees. Defaults to 0. A symbol in an `IconSequence` where `fixedRotation` is `false` is rotated relative to the angle of the edge on which it lies.

`scale`	**Type:** `number` The amount by which the symbol is scaled in size. For symbol markers, this defaults to 1; after scaling, the symbol may be of any size. For symbols on a polyline, this defaults to the stroke weight of the polyline; after scaling, the symbol must lie inside a square 22 pixels in size centered at the symbol's anchor.
`strokeColor`	**Type:** `string` The symbol's stroke color. All CSS3 colors are supported except for extended named colors. For symbol markers, this defaults to 'black'. For symbols on a polyline, this defaults to the stroke color of the polyline.
`strokeOpacity`	**Type:** `number` The symbol's stroke opacity. For symbol markers, this defaults to 1. For symbols on a polyline, this defaults to the stroke opacity of the polyline.
`strokeWeight`	**Type:** `number` The symbol's stroke weight. Defaults to the `scale` of the symbol.

SymbolPath class

`google.maps.SymbolPath` class

Built-in symbol paths.

Constant	
`BACKWARD_CLOSED_ARROW`	A backward-pointing closed arrow.
`BACKWARD_OPEN_ARROW`	A backward-pointing open arrow.
`CIRCLE`	A circle.
`FORWARD_CLOSED_ARROW`	A forward-pointing closed arrow.
`FORWARD_OPEN_ARROW`	A forward-pointing open arrow.

Animation class

`google.maps.Animation` class

Animations that can be played on a marker. Use the `setAnimation` method on Marker or the `animation` option to play an animation.

Constant	
`BOUNCE`	Marker bounces until animation is stopped.
`DROP`	Marker falls from the top of the map ending with a small bounce.

InfoWindow class

`google.maps.InfoWindow` class

An overlay that looks like a bubble and is often connected to a marker.

This class extends `MVCObject`.

Constructor	

`InfoWindow(`
`opts?:`*`InfoWindowOptions`*`)`
	Creates an info window with the given options. An InfoWindow can be placed on a map at a particular position or above a marker, depending on what is specified in the options. Unless auto-pan is disabled, an InfoWindow will pan the map to make itself visible when it is opened. After constructing an InfoWindow, you must call open to display it on the map. The user can click the close button on the InfoWindow to remove it from the map, or the developer can call close() for the same effect.

Methods	
`close()`	**Return Value:** None Closes this InfoWindow by removing it from the DOM structure.
`getContent()`	**Return Value:** `string|Node`
`getPosition()`	**Return Value:** `LatLng`
`getZIndex()`	**Return Value:** `number`

`open(map?:Map|`
`StreetViewPanorama,`
	Return Value: None
	Opens this InfoWindow on the given map. Optionally, an InfoWindow

anchor?:MVCObject)	can be associated with an anchor. In the core API, the only anchor is the Marker class. However, an anchor can be any MVCObject that exposes a LatLng `position` property and optionally a Point `anchorPoint` property for calculating the `pixelOffset` (see InfoWindowOptions). The `anchorPoint` is the offset from the anchor's position to the tip of the InfoWindow.
setContent(content:string|Node)	**Return Value:** None
setOptions(options:*InfoWindowOptions*)	**Return Value:** None
setPosition(position:LatLng|LatLngLiteral)	**Return Value:** None
setZIndex(zIndex:number)	**Return Value:** None

Events	
closeclick	**Arguments:** None This event is fired when the close button was clicked.
content_changed	**Arguments:** None This event is fired when the content property changes.
domready	**Arguments:** None This event is fired when the `<div>` containing the **`InfoWindow`**'s content is attached to the DOM. You may wish to monitor this event if you are building out your info window content dynamically.
position_changed	**Arguments:** None This event is fired when the position property changes.
zindex_changed	**Arguments:** None This event is fired when the InfoWindow's zIndex changes.

InfoWindowOptions object specification

`google.maps.InfoWindowOptions` object specification

Properties	
`content`	**Type:** `string|Node` Content to display in the InfoWindow. This can be an HTML element, a plain-text string, or a string containing HTML. The InfoWindow will be sized according to the content. To set an explicit size for the content, set content to be a HTML element with that size.
`disableAutoPan`	**Type:** `boolean` Disable auto-pan on open. By default, the info window will pan the map so that it is fully visible when it opens.
`maxWidth`	**Type:** `number` Maximum width of the infowindow, regardless of content's width. This value is only considered if it is set before a call to open. To change the maximum width when changing content, call close, setOptions, and then open.
`pixelOffset`	**Type:** `Size` The offset, in pixels, of the tip of the info window from the point on the map at whose geographical coordinates the info window is anchored. If an InfoWindow is opened with an anchor, the `pixelOffset` will be calculated from the anchor's `anchorPoint` property.
`position`	**Type:** `LatLng|LatLngLiteral` The LatLng at which to display this InfoWindow. If the InfoWindow is opened with an anchor, the anchor's position will be used instead.
`zIndex`	**Type:** `number` All InfoWindows are displayed on the map in order of their zIndex, with higher values displaying in front of InfoWindows with lower values. By default, InfoWindows are displayed according to their latitude, with InfoWindows of lower latitudes appearing in front of InfoWindows at higher latitudes. InfoWindows are always displayed in front of markers.

Polyline class

`google.maps.Polyline` class

A polyline is a linear overlay of connected line segments on the map.

This class extends `MVCObject`.

Constructor	
`Polyline(opts?:PolylineOptions)`	Create a polyline using the passed `PolylineOptions`, which specify both the path of the polyline and the stroke style to use when drawing the polyline. You may pass either an array of `LatLng`s or an `MVCArray` of `LatLng`s when constructing a polyline, though simple arrays are converted to `MVCArray`s within the polyline upon instantiation.

Methods	
`getDraggable()`	**Return Value:** `boolean` Returns whether this shape can be dragged by the user.
`getEditable()`	**Return Value:** `boolean` Returns whether this shape can be edited by the user.
`getMap()`	**Return Value:** `Map` Returns the map on which this shape is attached.
`getPath()`	**Return Value:** `MVCArray<LatLng>` Retrieves the path.
`getVisible()`	**Return Value:** `boolean` Returns whether this poly is visible on the map.
`setDraggable(draggable:boolean)`	**Return Value:** None If set to true, the user can drag this shape over the map. The `geodesic` property defines the mode of dragging.
`setEditable(editable:boolean)`	**Return Value:** None If set to true, the user can edit this shape by dragging the control points shown at the vertices and on each segment.
`setMap(map:Map)`	**Return Value:** None Renders this shape on the specified map. If map is set to null, the shape will be removed.
`setOptions(options:PolylineOptions)`	**Return Value:** None
`setPath(path:MVCArray<LatLng>|`	**Return Value:** None

`setPath(path:MVCArray<LatLng>|Array<LatLng|LatLngLiteral>)`	**Return Value:** None Sets the path. See `PolylineOptions` for more details.
`setVisible(visible:boolean)`	**Return Value:** None Hides this poly if set to `false`.

Events	
`click`	**Arguments:** *`PolyMouseEvent`* This event is fired when the DOM click event is fired on the Polyline.
`dblclick`	**Arguments:** *`PolyMouseEvent`* This event is fired when the DOM dblclick event is fired on the Polyline.
`drag`	**Arguments:** *`MouseEvent`* This event is repeatedly fired while the user drags the polyline.
`dragend`	**Arguments:** *`MouseEvent`* This event is fired when the user stops dragging the polyline.
`dragstart`	**Arguments:** *`MouseEvent`* This event is fired when the user starts dragging the polyline.
`mousedown`	**Arguments:** *`PolyMouseEvent`* This event is fired when the DOM mousedown event is fired on the Polyline.
`mousemove`	**Arguments:** *`PolyMouseEvent`* This event is fired when the DOM mousemove event is fired on the Polyline.
`mouseout`	**Arguments:** *`PolyMouseEvent`* This event is fired on Polyline mouseout.
`mouseover`	**Arguments:** *`PolyMouseEvent`* This event is fired on Polyline mouseover.
`mouseup`	**Arguments:** *`PolyMouseEvent`* This event is fired when the DOM mouseup event is fired on the Polyline.
`rightclick`	**Arguments:** *`PolyMouseEvent`* This event is fired when the Polyline is right-clicked on.

PolylineOptions object specification

`google.maps.PolylineOptions` object specification

Properties	
`clickable`	**Type:** `boolean` Indicates whether this `Polyline` handles mouse events. Defaults to `true`.
`draggable`	**Type:** `boolean` If set to true, the user can drag this shape over the map. The `geodesic` property defines the mode of dragging. Defaults to `false`.
`editable`	**Type:** `boolean` If set to true, the user can edit this shape by dragging the control points shown at the vertices and on each segment. Defaults to `false`.
`geodesic`	**Type:** `boolean` When true, edges of the polygon are interpreted as geodesic and will follow the curvature of the Earth. When false, edges of the polygon are rendered as straight lines in screen space. Note that the shape of a geodesic polygon may appear to change when dragged, as the dimensions are maintained relative to the surface of the earth. Defaults to `false`.
`icons`	**Type:** `Array<IconSequence>` The icons to be rendered along the polyline.
`map`	**Type:** `Map` Map on which to display Polyline.
`path`	**Type:** `MVCArray<LatLng>|Array<LatLng|LatLngLiteral>` The ordered sequence of coordinates of the Polyline. This path may be specified using either a simple array of `LatLng`s, or an `MVCArray` of `LatLng`s. Note that if you pass a simple array, it will be converted to an `MVCArray` Inserting or removing LatLngs in the `MVCArray` will automatically update the polyline on the map.
`strokeColor`	**Type:** `string` The stroke color. All CSS3 colors are supported except for extended named colors.
`strokeOpacity`	**Type:** `number`

	The stroke opacity between 0.0 and 1.0.
`strokeWeight`	**Type:** `number` The stroke width in pixels.
`visible`	**Type:** `boolean` Whether this polyline is visible on the map. Defaults to `true`.
`zIndex`	**Type:** `number` The zIndex compared to other polys.

IconSequence object specification

`google.maps.IconSequence` object specification

Describes how icons are to be rendered on a line.

If your polyline is geodesic, then the distances specified for both offset and repeat are calculated in meters by default. Setting either offset or repeat to a pixel value will cause the distances to be calculated in pixels on the screen.

Properties	
`fixedRotation`	**Type:** `boolean` If `true`, each icon in the sequence has the same fixed rotation regardless of the angle of the edge on which it lies. Defaults to `false`, in which case each icon in the sequence is rotated to align with its edge.
`icon`	**Type:** `Symbol` The icon to render on the line.
`offset`	**Type:** `string` The distance from the start of the line at which an icon is to be rendered. This distance may be expressed as a percentage of line's length (e.g. '50%') or in pixels (e.g. '50px'). Defaults to '100%'.
`repeat`	**Type:** `string` The distance between consecutive icons on the line. This distance may be expressed as a percentage of the line's length (e.g. '50%') or in pixels (e.g. '50px'). To disable repeating of the icon, specify '0'. Defaults to '0'.

Polygon class

`google.maps.Polygon` class

A polygon (like a polyline) defines a series of connected coordinates in an ordered sequence; additionally, polygons form a closed loop and define a filled region.

This class extends `MVCObject`.

Constructor	
`Polygon(opts?:`*`PolygonOptions`*`)`	Create a polygon using the passed *`PolygonOptions`*, which specify the polygon's path, the stroke style for the polygon's edges, and the fill style for the polygon's interior regions. A polygon may contain one or more paths, where each path consists of an array of `LatLng`s. You may pass either an array of LatLngs or an `MVCArray` of `LatLng`s when constructing these paths. Arrays are converted to `MVCArray`s within the polygon upon instantiation.

Methods	
`getDraggable()`	**Return Value:** `boolean` Returns whether this shape can be dragged by the user.
`getEditable()`	**Return Value:** `boolean` Returns whether this shape can be edited by the user.
`getMap()`	**Return Value:** `Map` Returns the map on which this shape is attached.
`getPath()`	**Return Value:** `MVCArray<LatLng>` Retrieves the first path.
`getPaths()`	**Return Value:** `MVCArray<MVCArray<LatLng>>` Retrieves the paths for this polygon.
`getVisible()`	**Return Value:** `boolean`

`getVisible()`	**Return Value:** `boolean` Returns whether this poly is visible on the map.
`setDraggable(draggable:boolean)`	**Return Value:** None If set to true, the user can drag this shape over the map. The `geodesic` property defines the mode of dragging.
`setEditable(editable:boolean)`	**Return Value:** None If set to true, the user can edit this shape by dragging the control points shown at the vertices and on each segment.
`setMap(map:Map)`	**Return Value:** None Renders this shape on the specified map. If map is set to null, the shape will be removed.
`setOptions(options:`*`PolygonOptions`*`)`	**Return Value:** None
`setPath(path:MVCArray<LatLng>|Array<LatLng|LatLngLiteral>)`	**Return Value:** None Sets the first path. See `PolygonOptions` for more details.
`setPaths(paths:MVCArray<MVCArray<LatLng>>|MVCArray<LatLng>|Array<Array<LatLng|LatLngLiteral>>|Array<LatLng|LatLngLiteral>)`	**Return Value:** None Sets the path for this polygon.
`setVisible(visible:boolean)`	**Return Value:** None Hides this poly if set to `false`.

Events

`click`	**Arguments:** *`PolyMouseEvent`* This event is fired when the DOM click event is fired on the Polygon.
`dblclick`	**Arguments:** *`PolyMouseEvent`* This event is fired when the DOM dblclick event is fired on the Polygon.
`drag`	**Arguments:** *`MouseEvent`* This event is repeatedly fired while the user drags the polygon.

`dragend`	**Arguments:** *`MouseEvent`* This event is fired when the user stops dragging the polygon.
`dragstart`	**Arguments:** *`MouseEvent`* This event is fired when the user starts dragging the polygon.
`mousedown`	**Arguments:** *`PolyMouseEvent`* This event is fired when the DOM mousedown event is fired on the Polygon.
`mousemove`	**Arguments:** *`PolyMouseEvent`* This event is fired when the DOM mousemove event is fired on the Polygon.
`mouseout`	**Arguments:** *`PolyMouseEvent`* This event is fired on Polygon mouseout.
`mouseover`	**Arguments:** *`PolyMouseEvent`* This event is fired on Polygon mouseover.
`mouseup`	**Arguments:** *`PolyMouseEvent`* This event is fired when the DOM mouseup event is fired on the Polygon.
`rightclick`	**Arguments:** *`PolyMouseEvent`* This event is fired when the Polygon is right-clicked on.

PolygonOptions object specification

`google.maps.PolygonOptions` object specification

Properties	
`clickable`	**Type:** `boolean` Indicates whether this `Polygon` handles mouse events. Defaults to `true`.
`draggable`	**Type:** `boolean` If set to true, the user can drag this shape over the map. The `geodesic` property defines
`editable`	**Type:** `boolean` If set to true, the user can edit this shape by dragging the control points shown at the ve

fillColor	**Type:** string The fill color. All CSS3 colors are supported except for extended named colors.
fillOpacity	**Type:** number The fill opacity between 0.0 and 1.0
geodesic	**Type:** boolean When true, edges of the polygon are interpreted as geodesic and will follow the curvatur rendered as straight lines in screen space. Note that the shape of a geodesic polygon m dimensions are maintained relative to the surface of the earth. Defaults to `false`.
map	**Type:** Map Map on which to display Polygon.
paths	**Type:** MVCArray<MVCArray<LatLng>>|MVCArray<LatLng>|Array<Array<LatLng|Lat The ordered sequence of coordinates that designates a closed loop. Unlike polylines, a p result, the paths property may specify one or more arrays of `LatLng` coordinates. Paths vertex of the path as the last vertex. Simple polygons may be defined using a single arra an array of arrays. Any simple arrays are converted into MVCArrays. Inserting or removi update the polygon on the map.
strokeColor	**Type:** string The stroke color. All CSS3 colors are supported except for extended named colors.
strokeOpacity	**Type:** number The stroke opacity between 0.0 and 1.0
strokePosition	**Type:** StrokePosition The stroke position. Defaults to CENTER. This property is not supported on Internet Expl
strokeWeight	**Type:** number The stroke width in pixels.
visible	**Type:** boolean Whether this polygon is visible on the map. Defaults to `true`.
zIndex	**Type:** number The zIndex compared to other polys.

PolyMouseEvent object specification

`google.maps.PolyMouseEvent` object specification

This object is returned from mouse events on polylines and polygons.

This object extends `MouseEvent`.

Properties	
`edge`	**Type:** `number` The index of the edge within the path beneath the cursor when the event occurred, if the event occurred on a mid-point on an editable polygon.
`path`	**Type:** `number` The index of the path beneath the cursor when the event occurred, if the event occurred on a vertex and the polygon is editable. Otherwise undefined.
`vertex`	**Type:** `number` The index of the vertex beneath the cursor when the event occurred, if the event occurred on a vertex and the polyline or polygon is editable. If the event does not occur on a vertex, the value is undefined.

Rectangle class

`google.maps.Rectangle` class

A rectangle overlay.

This class extends `MVCObject`.

Constructor	

`Rectangle(`
`opts?:`*`RectangleOptions`*`)`
	Create a rectangle using the passed *`RectangleOptions`*, which specify the bounds and style.

Methods	
`getBounds()`	**Return Value:** `LatLngBounds` Returns the bounds of this rectangle.

	Returns the bounds of this rectangle.
`getDraggable()`	**Return Value:** `boolean` Returns whether this rectangle can be dragged by the user.
`getEditable()`	**Return Value:** `boolean` Returns whether this rectangle can be edited by the user.
`getMap()`	**Return Value:** `Map` Returns the map on which this rectangle is displayed.
`getVisible()`	**Return Value:** `boolean` Returns whether this rectangle is visible on the map.
`setBounds(bounds:LatLngBounds)`	**Return Value:** None Sets the bounds of this rectangle.
`setDraggable(draggable:boolean)`	**Return Value:** None If set to true, the user can drag this rectangle over the map.
`setEditable(editable:boolean)`	**Return Value:** None If set to true, the user can edit this rectangle by dragging the control points shown at the corners and on each edge.
`setMap(map:Map)`	**Return Value:** None Renders the rectangle on the specified map. If map is set to null, the rectangle will be removed.
`setOptions(options:`*`RectangleOptions`*`)`	**Return Value:** None
`setVisible(visible:boolean)`	**Return Value:** None Hides this rectangle if set to `false`.

Events	
`bounds_changed`	**Arguments:** None This event is fired when the rectangle's bounds are changed.
`click`	**Arguments:** *`MouseEvent`* This event is fired when the DOM click event is fired on the rectangle.

`dblclick`	**Arguments:** *[MouseEvent]* This event is fired when the DOM dblclick event is fired on the rectangle.
`drag`	**Arguments:** *[MouseEvent]* This event is repeatedly fired while the user drags the rectangle.
`dragend`	**Arguments:** *[MouseEvent]* This event is fired when the user stops dragging the rectangle.
`dragstart`	**Arguments:** *[MouseEvent]* This event is fired when the user starts dragging the rectangle.
`mousedown`	**Arguments:** *[MouseEvent]* This event is fired when the DOM mousedown event is fired on the rectangle.
`mousemove`	**Arguments:** *[MouseEvent]* This event is fired when the DOM mousemove event is fired on the rectangle.
`mouseout`	**Arguments:** *[MouseEvent]* This event is fired on rectangle mouseout.
`mouseover`	**Arguments:** *[MouseEvent]* This event is fired on rectangle mouseover.
`mouseup`	**Arguments:** *[MouseEvent]* This event is fired when the DOM mouseup event is fired on the rectangle.
`rightclick`	**Arguments:** *[MouseEvent]* This event is fired when the rectangle is right-clicked on.

RectangleOptions object specification

`google.maps.RectangleOptions` object specification

Properties	
`bounds`	**Type:** `LatLngBounds` The bounds.

`clickable`	**Type:** `boolean` Indicates whether this `Rectangle` handles mouse events. Defaults to `true`.
`draggable`	**Type:** `boolean` If set to true, the user can drag this rectangle over the map. Defaults to `false`.
`editable`	**Type:** `boolean` If set to true, the user can edit this rectangle by dragging the control points shown at the corners and on each edge. Defaults to `false`.
`fillColor`	**Type:** `string` The fill color. All CSS3 colors are supported except for extended named colors.
`fillOpacity`	**Type:** `number` The fill opacity between 0.0 and 1.0
`map`	**Type:** `Map` Map on which to display Rectangle.
`strokeColor`	**Type:** `string` The stroke color. All CSS3 colors are supported except for extended named colors.
`strokeOpacity`	**Type:** `number` The stroke opacity between 0.0 and 1.0
`strokePosition`	**Type:** `StrokePosition` The stroke position. Defaults to CENTER. This property is not supported on Internet Explorer 8 and earlier.
`strokeWeight`	**Type:** `number` The stroke width in pixels.
`visible`	**Type:** `boolean` Whether this rectangle is visible on the map. Defaults to `true`.
`zIndex`	**Type:** `number` The zIndex compared to other polys.

Circle class

`google.maps.Circle` class

A circle on the Earth's surface; also known as a "spherical cap".

This class extends `MVCObject`.

Constructor	

`Circle(`
`opts?:`*`CircleOptions`*`)`
	Create a circle using the passed *`CircleOptions`*, which specify the center, radius, and style.

Methods	
`getBounds()`	**Return Value:** `LatLngBounds` Gets the `LatLngBounds` of this Circle.
`getCenter()`	**Return Value:** `LatLng` Returns the center of this circle.
`getDraggable()`	**Return Value:** `boolean` Returns whether this circle can be dragged by the user.
`getEditable()`	**Return Value:** `boolean` Returns whether this circle can be edited by the user.
`getMap()`	**Return Value:** `Map` Returns the map on which this circle is displayed.
`getRadius()`	**Return Value:** `number` Returns the radius of this circle (in meters).
`getVisible()`	**Return Value:** `boolean` Returns whether this circle is visible on the map.

`setCenter(`
`center:LatLng|`
`LatLngLiteral)`
	Return Value: None
	Sets the center of this circle.

`setDraggable(`
`draggable:boolean)`
	Return Value: None

	If set to true, the user can drag this circle over the map.
setEditable(editable:boolean)	**Return Value:** None If set to true, the user can edit this circle by dragging the control points shown at the center and around the circumference of the circle.
setMap(map:Map)	**Return Value:** None Renders the circle on the specified map. If map is set to null, the circle will be removed.
setOptions(options:*CircleOptions*)	**Return Value:** None
setRadius(radius:number)	**Return Value:** None Sets the radius of this circle (in meters).
setVisible(visible:boolean)	**Return Value:** None Hides this circle if set to `false`.

Events	
center_changed	**Arguments:** None This event is fired when the circle's center is changed.
click	**Arguments:** *MouseEvent* This event is fired when the DOM click event is fired on the circle.
dblclick	**Arguments:** *MouseEvent* This event is fired when the DOM dblclick event is fired on the circle.
drag	**Arguments:** *MouseEvent* This event is repeatedly fired while the user drags the circle.
dragend	**Arguments:** *MouseEvent* This event is fired when the user stops dragging the circle.
dragstart	**Arguments:** *MouseEvent* This event is fired when the user starts dragging the circle.
mousedown	**Arguments:** *MouseEvent* This event is fired when the DOM mousedown event is fired on the circle.

`mousemove`	**Arguments:** *`MouseEvent`* This event is fired when the DOM mousemove event is fired on the circle.
`mouseout`	**Arguments:** *`MouseEvent`* This event is fired on circle mouseout.
`mouseover`	**Arguments:** *`MouseEvent`* This event is fired on circle mouseover.
`mouseup`	**Arguments:** *`MouseEvent`* This event is fired when the DOM mouseup event is fired on the circle.
`radius_changed`	**Arguments:** None This event is fired when the circle's radius is changed.
`rightclick`	**Arguments:** *`MouseEvent`* This event is fired when the circle is right-clicked on.

CircleOptions object specification

`google.maps.CircleOptions` object specification

Properties	
`center`	**Type:** `LatLng` The center
`clickable`	**Type:** `boolean` Indicates whether this `Circle` handles mouse events. Defaults to `true`.
`draggable`	**Type:** `boolean` If set to true, the user can drag this circle over the map. Defaults to `false`.
`editable`	**Type:** `boolean` If set to true, the user can edit this circle by dragging the control points shown at the center and around the circumference of the circle. Defaults to `false`.
`fillColor`	**Type:** `string` The fill color. All CSS3 colors are supported except for extended named colors.

	The fill color. All CSS3 colors are supported except for extended named colors.
`fillOpacity`	**Type:** `number` The fill opacity between 0.0 and 1.0
`map`	**Type:** `Map` Map on which to display Circle.
`radius`	**Type:** `number` The radius in meters on the Earth's surface
`strokeColor`	**Type:** `string` The stroke color. All CSS3 colors are supported except for extended named colors.
`strokeOpacity`	**Type:** `number` The stroke opacity between 0.0 and 1.0
`strokePosition`	**Type:** `StrokePosition` The stroke position. Defaults to CENTER. This property is not supported on Internet Explorer 8 and earlier.
`strokeWeight`	**Type:** `number` The stroke width in pixels.
`visible`	**Type:** `boolean` Whether this circle is visible on the map. Defaults to `true`.
`zIndex`	**Type:** `number` The zIndex compared to other polys.

StrokePosition object specification

`google.maps.StrokePosition` object specification

The possible positions of the stroke on a polygon.

Constant	
`CENTER`	The stroke is centered on the polygon's path, with half the stroke inside the polygon and half the stroke outside the polygon.

`INSIDE`	The stroke lies inside the polygon.
`OUTSIDE`	The stroke lies outside the polygon.

GroundOverlay class

`google.maps.GroundOverlay` class

A rectangular image overlay on the map.

This class extends `MVCObject`.

Constructor	
`GroundOverlay(url:string, bounds:LatLngBounds, opts?:`*`GroundOverlayOptions`*`)`	Creates a ground overlay from the provided image URL and its `LatLngBounds`. The image is scaled to fit the current bounds, and projected using the current map projection.

Methods	
`getBounds()`	**Return Value:** `LatLngBounds` Gets the `LatLngBounds` of this overlay.
`getMap()`	**Return Value:** `Map` Returns the map on which this ground overlay is displayed.
`getOpacity()`	**Return Value:** `number` Returns the opacity of this ground overlay.
`getUrl()`	**Return Value:** `string` Gets the url of the projected image.
`setMap(map:Map)`	**Return Value:** None Renders the ground overlay on the specified map. If map is set to null, the overlay is removed.
`setOpacity(opacity:number)`	**Return Value:** None Sets the opacity of this ground overlay.

Events	
click	**Arguments:** *MouseEvent* This event is fired when the DOM click event is fired on the `GroundOverlay`.
dblclick	**Arguments:** *MouseEvent* This event is fired when the DOM dblclick event is fired on the `GroundOverlay`.

GroundOverlayOptions object specification

`google.maps.GroundOverlayOptions` object specification

This object defines the properties that can be set on a `GroundOverlay` object.

Properties	
clickable	**Type:** boolean If `true`, the ground overlay can receive mouse events.
map	**Type:** Map The map on which to display the overlay.
opacity	**Type:** number The opacity of the overlay, expressed as a number between 0 and 1. Optional. Defaults to 1.

OverlayView class

`google.maps.OverlayView` class

You can implement this class if you want to display custom types of overlay objects on the map.

Inherit from this class by setting your overlay's prototype: `MyOverlay.prototype = new google.maps.OverlayView();`. The `OverlayView` constructor is guaranteed to be an empty function.

You must implement three methods: `onAdd()`, `draw()`, and `onRemove()`.

- In the `onAdd()` method, you should create DOM objects and append them as children of the panes.
- In the `draw()` method, you should position these elements.
- In the `onRemove()` method, you should remove the objects from the DOM.

You must call `setMap()` with a valid `Map` object to trigger the call to the `onAdd()` method and `setMap(null)` in order to trigger the `onRemove()` method. The `setMap()` method can be called at the time of construction or at any point afterward when the overlay should be re-shown after removing. The `draw()` method will then be called whenever a map property changes that could change the position of the element, such as zoom, center, or map type.

This class extends `MVCObject`.

Constructor	
`OverlayView()`	Creates an `OverlayView`.

Methods	
`draw()`	**Return Value:** None Implement this method to draw or update the overlay. This method is called after onAdd() and when the position from projection.fromLatLngToPixel() would return a new value for a given LatLng. This can happen on change of zoom, center, or map type. It is not necessarily called on drag or resize.
`getMap()`	**Return Value:** `Map`|`StreetViewPanorama`
`getPanes()`	**Return Value:** `MapPanes` Returns the panes in which this OverlayView can be rendered. The panes are not initialized until onAdd is called by the API.
`getProjection()`	**Return Value:** `MapCanvasProjection` Returns the `MapCanvasProjection` object associated with this `OverlayView`. The projection is not initialized until onAdd is called by the API.
`onAdd()`	**Return Value:** None Implement this method to initialize the overlay DOM elements. This method is called once after setMap() is called with a valid map. At this point, panes and projection will have been initialized.

`onRemove()`	**Return Value:** None Implement this method to remove your elements from the DOM. This method is called once following a call to setMap(null).
`setMap(map:Map|StreetViewPanorama)`	**Return Value:** None Adds the overlay to the map or panorama.

MapPanes object specification

`google.maps.MapPanes` object specification

This object contains the DOM elements in which overlays are rendered. They are listed below with 'Pane 0' at the bottom and 'Pane 4' at the top.

Properties	
`floatPane`	**Type:** Node This pane contains the info window. It is above all map overlays. (Pane 4).
`mapPane`	**Type:** Node This pane is the lowest pane and is above the tiles. It may not receive DOM events. (Pane 0).
`markerLayer`	**Type:** Node This pane contains markers. It may not receive DOM events. (Pane 2).
`overlayLayer`	**Type:** Node This pane contains polylines, polygons, ground overlays and tile layer overlays. It may not receive DOM events. (Pane 1).
`overlayMouseTarget`	**Type:** Node This pane contains elements that receive DOM events. (Pane 3).

MapCanvasProjection object specification

`google.maps.MapCanvasProjection` object specification

This object is made available to the `OverlayView` from within the draw method. It is not guaranteed to be initialized until draw is called.

This object extends `MVCObject`.

Methods	
`fromContainerPixelToLatLng(pixel:Point, nowrap?:boolean)`	**Return Value:** `LatLng` Computes the geographical coordinates from pixel coordinates in the map's container.
`fromDivPixelToLatLng(pixel:Point, nowrap?:boolean)`	**Return Value:** `LatLng` Computes the geographical coordinates from pixel coordinates in the div that holds the draggable map.
`fromLatLngToContainerPixel(latLng:LatLng)`	**Return Value:** `Point` Computes the pixel coordinates of the given geographical location in the map's container element.
`fromLatLngToDivPixel(latLng:LatLng)`	**Return Value:** `Point` Computes the pixel coordinates of the given geographical location in the DOM element that holds the draggable map.
`getWorldWidth()`	**Return Value:** `number` The width of the world in pixels in the current zoom level. For projections with a heading angle of either 90 or 270 degrees, this corresponds to the pixel span in the Y-axis.

Geocoder class

`google.maps.Geocoder` class

A service for converting between an address and a `LatLng`.

Constructor	
`Geocoder()`	Creates a new instance of a `Geocoder` that sends geocode requests to Google servers.

Methods

`geocode(request:GeocoderRequest, callback:function(Array<GeocoderResult>, GeocoderStatus))`	**Return Value:** None Geocode a request.

GeocoderRequest object specification

`google.maps.GeocoderRequest` object specification

The specification for a geocoding request to be sent to the `Geocoder`.

Properties	
`address`	**Type:** `string` Address to geocode. One, and only one, of `address`, `location` and `placeId` must be supplied.
`bounds`	**Type:** `LatLngBounds` `LatLngBounds` within which to search. Optional.
`componentRestrictions`	**Type:** `GeocoderComponentRestrictions` Components are used to restrict results to a specific area. A filter consists of one or more of: `route`, `locality`, `administrativeArea`, `postalCode`, `country`. Only the results that match all the filters will be returned. Filter values support the same methods of spelling correction and partial matching as other geocoding requests. Optional.
`location`	**Type:** `LatLng`|`LatLngLiteral` `LatLng` (or `LatLngLiteral`) for which to search. The geocoder performs a reverse geocode. See Reverse Geocoding for more information. One, and only one, of `address`, `location` and `placeId` must be supplied.
`placeId`	**Type:** `string` The place ID associated with the location. Place IDs uniquely identify a place in the Google Places database and on Google Maps. Learn more about place IDs in the Places API developer guide. The geocoder performs a reverse geocode. See Reverse Geocoding for more information. One, and only one, of `address`, `location` and `placeId` must be supplied.
`region`	**Type:** `string`

	Country code used to bias the search, specified as a Unicode region subtag / CLDR identifier. Optional.

GeocoderComponentRestrictions object specification

`google.maps.GeocoderComponentRestrictions` object specification

`GeocoderComponentRestrictions` represents a set of filters that resolve to a specific area. For details on how this works, see Geocoding Component Filtering.

Properties	
`administrativeArea`	**Type:** `string` Matches all the `administrative_area levels`. Optional.
`country`	**Type:** `string` Matches a country name or a two letter ISO 3166-1 country code. Optional.
`locality`	**Type:** `string` Matches against both `locality` and `sublocality` types. Optional.
`postalCode`	**Type:** `string` Matches `postal_code` and `postal_code_prefix`. Optional.
`route`	**Type:** `string` Matches the long or short name of a `route`. Optional.

GeocoderStatus class

`google.maps.GeocoderStatus` class

The status returned by the `Geocoder` on the completion of a call to `geocode()`.

Constant	
`ERROR`	There was a problem contacting the Google servers.
`INVALID_REQUEST`	This `GeocoderRequest` was invalid.

`OK`	The response contains a valid `GeocoderResponse`.
`OVER_QUERY_LIMIT`	The webpage has gone over the requests limit in too short a period of time.
`REQUEST_DENIED`	The webpage is not allowed to use the geocoder.
`UNKNOWN_ERROR`	A geocoding request could not be processed due to a server error. The request may succeed if you try again.
`ZERO_RESULTS`	No result was found for this `GeocoderRequest`.

GeocoderResult object specification

`google.maps.GeocoderResult` object specification

A single geocoder result retrieved from the geocode server. A geocode request may return multiple result objects. Note that though this result is "JSON-like," it is not strictly JSON, as it indirectly includes a `LatLng` object.

Properties	
`address_components`	**Type:** `Array<GeocoderAddressComponent>` An array of `GeocoderAddressComponents`
`formatted_address`	**Type:** `string` A string containing the human-readable address of this location.
`geometry`	**Type:** `GeocoderGeometry` A `GeocoderGeometry` object
`partial_match`	**Type:** `boolean` Whether the geocoder did not return an exact match for the original request, though it was able to match part of the requested address.
`place_id`	**Type:** `string` The place ID associated with the location. Place IDs uniquely identify a place in the Google Places database and on Google Maps. Learn more about Place IDs in the Places API developer guide.
`postcode_localities`	**Type:** `Array<string>` An array of strings denoting all the localities contained in a postal code. This is only present when the result is a postal code that contains multiple localities.

`types`	**Type:** `Array<string>` An array of strings denoting the type of the returned geocoded element. For a list of possible strings, refer to the Address Component Types section of the Developer's Guide.

GeocoderAddressComponent object specification

`google.maps.GeocoderAddressComponent` object specification

A single address component within a `GeocoderResult`. A full address may consist of multiple address components.

Properties	
`long_name`	**Type:** `string` The full text of the address component
`short_name`	**Type:** `string` The abbreviated, short text of the given address component
`types`	**Type:** `Array<string>` An array of strings denoting the type of this address component. A list of valid types can be found here

GeocoderGeometry object specification

`google.maps.GeocoderGeometry` object specification

Geometry information about this `GeocoderResult`

Properties	
`bounds`	**Type:** `LatLngBounds` The precise bounds of this `GeocoderResult`, if applicable
`location`	**Type:** `LatLng` The latitude/longitude coordinates of this result

	The latitude/longitude coordinates of this result
location_type	**Type:** GeocoderLocationType The type of location returned in location
viewport	**Type:** LatLngBounds The bounds of the recommended viewport for displaying this GeocoderResult

GeocoderLocationType class

google.maps.GeocoderLocationType class

Describes the type of location returned from a geocode.

Constant	
APPROXIMATE	The returned result is approximate.
GEOMETRIC_CENTER	The returned result is the geometric center of a result such a line (e.g. street) or polygon (region).
RANGE_INTERPOLATED	The returned result reflects an approximation (usually on a road) interpolated between two precise points (such as intersections). Interpolated results are generally returned when rooftop geocodes are unavailable for a street address.
ROOFTOP	The returned result reflects a precise geocode.

DirectionsRenderer class

google.maps.DirectionsRenderer class

Renders directions obtained from the DirectionsService.

This class extends MVCObject.

Constructor	
DirectionsRenderer(opts?:*DirectionsRendererOptions*)	Creates the renderer with the given options. Directions can be rendered on a map (as visual overlays) or additionally on a <div> panel (as textual instructions).

Methods	
getDirections()	**Return Value:** DirectionsResult Returns the renderer's current set of directions.
getMap()	**Return Value:** Map Returns the map on which the DirectionsResult is rendered.
getPanel()	**Return Value:** Node Returns the panel <div> in which the DirectionsResult is rendered.
getRouteIndex()	**Return Value:** number Returns the current (zero-based) route index in use by this DirectionsRenderer object.
setDirections(directions:*DirectionsResult*)	**Return Value:** None Set the renderer to use the result from the DirectionsService. Setting a valid set of directions in this manner will display the directions on the renderer's designated map and panel.
setMap(map:Map)	**Return Value:** None This method specifies the map on which directions will be rendered. Pass null to remove the directions from the map.
setOptions(options:*DirectionsRendererOptions*)	**Return Value:** None Change the options settings of this DirectionsRenderer after initialization.
setPanel(panel:Node)	**Return Value:** None This method renders the directions in a <div>. Pass null to remove the content from the panel.
setRouteIndex(routeIndex:number)	**Return Value:** None Set the (zero-based) index of the route in the DirectionsResult object to render. By default, the first route in the array will be rendered.

Events	
`directions_changed`	**Arguments:** None This event is fired when the rendered directions change, either when a new `DirectionsResult` is set or when the user finishes dragging a change to the directions path.

DirectionsRendererOptions object specification

`google.maps.DirectionsRendererOptions` object specification

This object defines the properties that can be set on a `DirectionsRenderer` object.

Properties	
`directions`	**Type:** `DirectionsResult` The directions to display on the map and/or in a `<div>` panel, retrieved as a `DirectionsResult` object from `DirectionsService`.
`draggable`	**Type:** `boolean` If true, allows the user to drag and modify the paths of routes rendered by this `DirectionsRenderer`.
`hideRouteList`	**Type:** `boolean` This property indicates whether the renderer should provide UI to select amongst alternative routes. By default, this flag is `false` and a user-selectable list of routes will be shown in the directions' associated panel. To hide that list, set `hideRouteList` to `true`.
`infoWindow`	**Type:** `InfoWindow` The `InfoWindow` in which to render text information when a marker is clicked. Existing info window content will be overwritten and its position moved. If no info window is specified, the `DirectionsRenderer` will create and use its own info window. This property will be ignored if `suppressInfoWindows` is set to true.
`map`	**Type:** `Map` Map on which to display the directions.
`markerOptions`	**Type:** `MarkerOptions` Options for the markers. All markers rendered by the

	Options for the markers. All markers rendered by the `DirectionsRenderer` will use these options.
`panel`	**Type:** `Node` The `<div>` in which to display the directions steps.
`polylineOptions`	**Type:** `PolylineOptions` Options for the polylines. All polylines rendered by the `DirectionsRenderer` will use these options.
`preserveViewport`	**Type:** `boolean` By default, the input map is centered and zoomed to the bounding box of this set of directions. If this option is set to `true`, the viewport is left unchanged, unless the map's center and zoom were never set.
`routeIndex`	**Type:** `number` The index of the route within the `DirectionsResult` object. The default value is 0.
`suppressBicyclingLayer`	**Type:** `boolean` Suppress the rendering of the `BicyclingLayer` when bicycling directions are requested.
`suppressInfoWindows`	**Type:** `boolean` Suppress the rendering of info windows.
`suppressMarkers`	**Type:** `boolean` Suppress the rendering of markers.
`suppressPolylines`	**Type:** `boolean` Suppress the rendering of polylines.

DirectionsService class

`google.maps.DirectionsService` class

A service for computing directions between two or more places.

Constructor	
`DirectionsService()`	Creates a new instance of a `DirectionsService` that sends directions queries to Google servers.

	to Google servers.

Methods	
`route(request:DirectionsRequest, callback:function(DirectionsResult, DirectionsStatus))`	**Return Value:** None Issue a directions search request.

DirectionsRequest object specification

`google.maps.DirectionsRequest` object specification

A directions query to be sent to the `DirectionsService`.

Properties	
`avoidFerries`	**Type:** `boolean` If true, instructs the Directions service to avoid ferries where possible. Optional.
`avoidHighways`	**Type:** `boolean` If true, instructs the Directions service to avoid highways where possible. Optional.
`avoidTolls`	**Type:** `boolean` If true, instructs the Directions service to avoid toll roads where possible. Optional.
`destination`	**Type:** `LatLng|string` Location of destination. This can be specified as either a string to be geocoded or a `LatLng`. Required.
`durationInTraffic`	**Type:** `boolean` Whether or not we should provide trip duration based on current traffic conditions. Only available to Maps API for Work customers.
`optimizeWaypoints`	**Type:** `boolean` If set to true, the `DirectionService` will attempt to re-order the supplied intermediate waypoints to minimize overall cost of the route. If waypoints are optimized, inspect

	DirectionsRoute.waypoint_order in the response to determine the new ordering.
origin	**Type:** LatLng|string Location of origin. This can be specified as either a string to be geocoded or a LatLng. Required.
provideRouteAlternatives	**Type:** boolean Whether or not route alternatives should be provided. Optional.
region	**Type:** string Region code used as a bias for geocoding requests. Optional.
transitOptions	**Type:** TransitOptions Settings that apply only to requests where travelMode is TRANSIT. This object will have no effect for other travel modes.
travelMode	**Type:** TravelMode Type of routing requested. Required.
unitSystem	**Type:** UnitSystem Preferred unit system to use when displaying distance. Defaults to the unit system used in the country of origin.
waypoints	**Type:** Array<DirectionsWaypoint> Array of intermediate waypoints. Directions will be calculated from the origin to the destination by way of each waypoint in this array. The maximum allowed waypoints is 8, plus the origin, and destination. Maps API for Work customers are allowed 23 waypoints, plus the origin, and destination. Waypoints are not supported for transit directions. Optional.

Place object specification

google.maps.Place object specification

Properties	
location	**Type:** LatLng|LatLngLiteral The LatLng of the entity described by this place. This must be provided for the Place to be considered valid.

`placeId`	**Type:** `string` The place ID of the place (such as a business or point of interest). The place ID is a unique identifier of a place in the Google Maps database. Note that the `placeId` is the most accurate way of identifying a place. If possible, you should specify the `placeId` rather than a `placeQuery`. A place ID can be retrieved from any request to the Places API, such as a TextSearch. Place IDs can also be retrieved from requests to the Geocoding API. For more information, see the overview of place IDs.
`query`	**Type:** `string` A search query describing the place (such as a business or point of interest). An example query is "Quay, Upper Level, Overseas Passenger Terminal 5 Hickson Road, The Rocks NSW". If possible, you should specify the `placeId` rather than a `placeQuery`. The API does not guarantee the accuracy of resolving the query string to a place. If both the `placeId` and `placeQuery` are provided, an error occurs.

TravelMode class

`google.maps.TravelMode` class

The valid travel modes that can be specified in a `DirectionsRequest` as well as the travel modes returned in a `DirectionsStep`.

Constant	
`BICYCLING`	Specifies a bicycling directions request.
`DRIVING`	Specifies a driving directions request.
`TRANSIT`	Specifies a transit directions request.
`WALKING`	Specifies a walking directions request.

UnitSystem class

`google.maps.UnitSystem` class

The valid unit systems that can be specified in a `DirectionsRequest`.

Constant	
`IMPERIAL`	Specifies that distances in the `DirectionsResult` should be expressed in imperial units.
`METRIC`	Specifies that distances in the `DirectionsResult` should be expressed in metric units.

TransitOptions object specification

`google.maps.TransitOptions` object specification

The TransitOptions object to be included in a `DirectionsRequest` when the travel mode is set to TRANSIT.

Properties	
`arrivalTime`	**Type:** `Date` The desired arrival time for the route, specified as a Date object. The Date object measures time in milliseconds since 1 January 1970. If arrival time is specified, departure time is ignored.
`departureTime`	**Type:** `Date` The desired departure time for the route, specified as a Date object. The Date object measures time in milliseconds since 1 January 1970. If neither departure time nor arrival time is specified, the time is assumed to be "now".
`modes`	**Type:** `Array<TransitMode>` One or more preferred modes of transit, such as bus or train. If no preference is given, the API returns the default best route.
`routingPreference`	**Type:** `TransitRoutePreference` A preference that can bias the choice of transit route, such as less walking. If no preference is given, the API returns the default best route.

TransitMode class

`google.maps.TransitMode` class

The valid transit mode e.g. bus that can be specified in a `TransitOptions`.

Constant	
BUS	Specifies bus as a preferred mode of transit.
RAIL	Specifies rail as a preferred mode of transit.
SUBWAY	Specifies subway as a preferred mode of transit.
TRAIN	Specifies train as a preferred mode of transit.
TRAM	Specifies tram as a preferred mode of transit.

TransitRoutePreference class

`google.maps.TransitRoutePreference` class

The valid transit route type that can be specified in a `TransitOptions`.

Constant	
FEWER_TRANSFERS	Specifies that the calculated route should prefer a limited number of transfers.
LESS_WALKING	Specifies that the calculated route should prefer limited amounts of walking.

TransitFare object specification

`google.maps.TransitFare` object specification

A fare of a `DirectionsRoute` consisting of value and currency.

DirectionsWaypoint object specification

`google.maps.DirectionsWaypoint` object specification

A `DirectionsWaypoint` represents a location between origin and destination through which the trip should be routed.

Properties	

`location`	**Type:** `LatLng|string` Waypoint location. Can be an address string or `LatLng`. Optional.
`stopover`	**Type:** `boolean` If `true`, indicates that this waypoint is a stop between the origin and destination. This has the effect of splitting the route into two. This value is `true` by default. Optional.

DirectionsStatus class

`google.maps.DirectionsStatus` class

The status returned by the `DirectionsService` on the completion of a call to `route()`.

Constant	
`INVALID_REQUEST`	The `DirectionsRequest` provided was invalid.
`MAX_WAYPOINTS_EXCEEDED`	Too many `DirectionsWaypoints` were provided in the `DirectionsRequest`. The total allowed waypoints is 8, plus the origin and destination. Maps API for Work customers are allowed 23 waypoints, plus the origin, and destination.
`NOT_FOUND`	At least one of the origin, destination, or waypoints could not be geocoded.
`OK`	The response contains a valid `DirectionsResult`.
`OVER_QUERY_LIMIT`	The webpage has gone over the requests limit in too short a period of time.
`REQUEST_DENIED`	The webpage is not allowed to use the directions service.
`UNKNOWN_ERROR`	A directions request could not be processed due to a server error. The request may succeed if you try again.
`ZERO_RESULTS`	No route could be found between the origin and destination.

DirectionsResult object specification

`google.maps.DirectionsResult` object specification

The directions response retrieved from the directions server. You can render these using a `DirectionsRenderer` or parse this object and render it yourself. You must display the warnings and copyrights as noted in the Maps API terms of service. Note that though this result is "JSON-like," it is not strictly JSON, as it indirectly includes `LatLng` objects.

Properties	
`geocoded_waypoints`	**Type:** `Array<DirectionsGeocodedWaypoint>` An array of `DirectionsGeocodedWaypoints`, each of which contains information about the geocoding of origin, destination and waypoints.
`routes`	**Type:** `Array<DirectionsRoute>` An array of `DirectionsRoutes`, each of which contains information about the legs and steps of which it is composed. There will only be one route unless the `DirectionsRequest` was made with `provideRouteAlternatives` set to `true`.

DirectionsGeocodedWaypoint object specification

`google.maps.DirectionsGeocodedWaypoint` object specification

A single geocoded waypoint.

Properties	
`partial_match`	**Type:** `boolean` Whether the geocoder did not return an exact match for the original waypoint, though it was able to match part of the requested address.
`place_id`	**Type:** `string` The place ID associated with the waypoint. Place IDs uniquely identify a place in the Google Places database and on Google Maps. Learn more about Place IDs in the Places API developer guide.
`types`	**Type:** `Array<string>` An array of strings denoting the type of the returned geocoded element. For a list of possible strings, refer to the Address Component Types section of the Developer's Guide.

DirectionsRoute object specification

`google.maps.DirectionsRoute` object specification

A single route containing a set of legs in a `DirectionsResult`. Note that though this object is "JSON-like," it is not strictly JSON, as it directly and indirectly includes `LatLng` objects.

Properties	
`bounds`	**Type:** `LatLngBounds` The bounds for this route.
`copyrights`	**Type:** `string` Copyrights text to be displayed for this route.
`fare`	**Type:** `TransitFare` The total fare for the whole transit trip. Only applicable to transit requests.
`legs`	**Type:** `Array<DirectionsLeg>` An array of `DirectionsLegs`, each of which contains information about the steps of which it is composed. There will be one leg for each waypoint or destination specified. So a route with no waypoints will contain one `DirectionsLeg` and a route with one waypoint will contain two.
`overview_path`	**Type:** `Array<LatLng>` An array of `LatLngs` representing the entire course of this route. The path is simplified in order to make it suitable in contexts where a small number of vertices is required (such as Static Maps API URLs).
`overview_polyline`	**Type:** `string` An encoded polyline representation of the route in overview_path. This polyline is an approximate (smoothed) path of the resulting directions.
`warnings`	**Type:** `Array<string>` Warnings to be displayed when showing these directions.
`waypoint_order`	**Type:** `Array<number>` If `optimizeWaypoints` was set to `true`, this field will contain the re-ordered permutation of the input waypoints. For example, if the input was: Origin: Los Angeles Waypoints: Dallas, Bangor, Phoenix Destination: New York and the optimized output was ordered as follows: Origin: Los Angeles Waypoints: Phoenix, Dallas, Bangor

	Destination: New York then this field will be an `Array` containing the values [2, 0, 1]. Note that the numbering of waypoints is zero-based. If any of the input waypoints has `stopover` set to `false`, this field will be empty, since route optimization is not available for such queries.

DirectionsLeg object specification

`google.maps.DirectionsLeg` object specification

A single leg consisting of a set of steps in a `DirectionsResult`. Some fields in the leg may not be returned for all requests. Note that though this result is "JSON-like," it is not strictly JSON, as it directly and indirectly includes `LatLng` objects.

Properties	
`arrival_time`	**Type:** `Time` An estimated arrival time for this leg. Only applicable for TRANSIT requests.
`departure_time`	**Type:** `Time` An estimated departure time for this leg. Only applicable for TRANSIT requests.
`distance`	**Type:** `Distance` The total distance covered by this leg. This property may be undefined as the distance may be unknown.
`duration`	**Type:** `Duration` The total duration of this leg. This property may be undefined as the duration may be unknown.
`duration_in_traffic`	**Type:** `Duration` The total duration of this leg, taking into account current traffic conditions. This property may be undefined as the duration may be unknown. Only available to Maps API for Work customers when `durationInTraffic` is set to `true` when making the request.
`end_address`	**Type:** `string` The address of the destination of this leg.
`end_location`	**Type:** `LatLng` The `DirectionsService` calculates directions between locations by using the

	The DirectionsService calculates directions between locations by using the nearest transportation option (usually a road) at the start and end locations. `end_location` indicates the actual geocoded destination, which may be different than the `end_location` of the last step if, for example, the road is not near the destination of this leg.
`start_address`	**Type:** `string` The address of the origin of this leg.
`start_location`	**Type:** `LatLng` The `DirectionsService` calculates directions between locations by using the nearest transportation option (usually a road) at the start and end locations. `start_location` indicates the actual geocoded origin, which may be different than the `start_location` of the first step if, for example, the road is not near the origin of this leg.
`steps`	**Type:** `Array<DirectionsStep>` An array of `DirectionsSteps`, each of which contains information about the individual steps in this leg.
`via_waypoints`	**Type:** `Array<LatLng>` An array of waypoints along this leg that were not specified in the original request, either as a result of a user dragging the polyline or selecting an alternate route.

DirectionsStep object specification

`google.maps.DirectionsStep` object specification

A single `DirectionsStep` in a `DirectionsResult`. Some fields may be undefined. Note that though this object is "JSON-like," it is not strictly JSON, as it directly includes `LatLng` objects.

Properties	
`distance`	**Type:** `Distance` The distance covered by this step. This property may be undefined as the distance may be unknown.
`duration`	**Type:** `Duration` The typical time required to perform this step in seconds and in text form. This

	property may be undefined as the duration may be unknown.
`end_location`	**Type:** `LatLng` The ending location of this step.
`instructions`	**Type:** `string` Instructions for this step.
`path`	**Type:** `Array<LatLng>` A sequence of `LatLngs` describing the course of this step.
`start_location`	**Type:** `LatLng` The starting location of this step.
`steps`	**Type:** `Array<DirectionsStep>` Sub-steps of this step. Specified for non-transit sections of transit routes.
`transit`	**Type:** `TransitDetails` Transit-specific details about this step. This property will be undefined unless the travel mode of this step is `TRANSIT`.
`travel_mode`	**Type:** `TravelMode` The mode of travel used in this step.

Distance object specification

`google.maps.Distance` object specification

A representation of distance as a numeric value and a display string.

Properties	
`text`	**Type:** `string` A string representation of the distance value, using the `UnitSystem` specified in the request.
`value`	**Type:** `number` The distance in meters.

Duration object specification

`google.maps.Duration` object specification

A representation of duration as a numeric value and a display string.

Properties	
`text`	**Type:** `string` A string representation of the duration value.
`value`	**Type:** `number` The duration in seconds.

Time object specification

`google.maps.Time` object specification

Properties	
`text`	**Type:** `string` A string representing the time's value. The time is displayed in the time zone of the transit stop.
`time_zone`	**Type:** `string` The time zone in which this stop lies. The value is the name of the time zone as defined in the IANA Time Zone Database, e.g. "America/New_York".
`value`	**Type:** `Date` The time of this departure or arrival, specified as a JavaScript Date object.

TransitDetails object specification

`google.maps.TransitDetails` object specification

Properties

arrival_stop	**Type:** TransitStop The arrival stop of this transit step.
arrival_time	**Type:** Time The arrival time of this step, specified as a Time object.
departure_stop	**Type:** TransitStop The departure stop of this transit step.
departure_time	**Type:** Time The departure time of this step, specified as a Time object.
headsign	**Type:** string The direction in which to travel on this line, as it is marked on the vehicle or at the departure stop.
headway	**Type:** number The expected number of seconds between equivalent vehicles at this stop.
line	**Type:** TransitLine Details about the transit line used in this step.
num_stops	**Type:** number The number of stops on this step. Includes the arrival stop, but not the departure stop.

TransitStop object specification

`google.maps.TransitStop` object specification

Properties	
location	**Type:** LatLng

	The location of this stop.
`name`	**Type:** `string` The name of this transit stop.

TransitLine object specification

`google.maps.TransitLine` object specification

Properties	
`agencies`	**Type:** `Array<TransitAgency>` The transit agency that operates this transit line.
`color`	**Type:** `string` The color commonly used in signage for this transit line, represented as a hex string.
`icon`	**Type:** `string` The URL for an icon associated with this line.
`name`	**Type:** `string` The full name of this transit line, e.g. "8 Avenue Local".
`short_name`	**Type:** `string` The short name of this transit line, e.g. "E".
`text_color`	**Type:** `string` The text color commonly used in signage for this transit line, represented as a hex string.
`url`	**Type:** `string` The agency's URL which is specific to this transit line.
`vehicle`	**Type:** `TransitVehicle` The type of vehicle used, e.g. train or bus.

TransitAgency object specification

`google.maps.TransitAgency` object specification

Properties	
`name`	**Type:** `string` The name of this transit agency.
`phone`	**Type:** `string` The transit agency's phone number.
`url`	**Type:** `string` The transit agency's URL.

TransitVehicle object specification

`google.maps.TransitVehicle` object specification

Properties	
`icon`	**Type:** `string` A URL for an icon that corresponds to the type of vehicle used on this line.
`local_icon`	**Type:** `string` A URL for an icon that corresponds to the type of vehicle used in this region instead of the more general icon.
`name`	**Type:** `string` A name for this type of TransitVehicle, e.g. "Train" or "Bus".
`type`	**Type:** `VehicleType` The type of vehicle used, e.g. train, bus, or ferry.

VehicleType object specification

`google.maps.VehicleType` object specification

Possible values for vehicle types. These values are specifed as strings, i.e. 'BUS' or 'TRAIN'.

Constant	
`BUS`	Bus.
`CABLE_CAR`	A vehicle that operates on a cable, usually on the ground. Aerial cable cars may be of the type `GONDOLA_LIFT`.
`COMMUTER_TRAIN`	Commuter rail.
`FERRY`	Ferry.
`FUNICULAR`	A vehicle that is pulled up a steep incline by a cable.
`GONDOLA_LIFT`	An aerial cable car.
`HEAVY_RAIL`	Heavy rail.
`HIGH_SPEED_TRAIN`	High speed train.
`INTERCITY_BUS`	Intercity bus.
`METRO_RAIL`	Light rail.
`MONORAIL`	Monorail.
`OTHER`	Other vehicles.
`RAIL`	Rail.
`SHARE_TAXI`	Share taxi is a sort of bus transport with ability to drop off and pick up passengers anywhere on its route. Generally share taxi uses minibus vehicles.
`SUBWAY`	Underground light rail.
`TRAM`	Above ground light rail.
`TROLLEYBUS`	Trolleybus.

ElevationService class

`google.maps.ElevationService` class

Defines a service class that talks directly to Google servers for requesting elevation data.

Constructor

`ElevationService()`	Creates a new instance of a `ElevationService` that sends elevation queries to Google servers.

Methods	
`getElevationAlongPath(request:PathElevationRequest, callback:function(Array<ElevationResult>, ElevationStatus))`	**Return Value:** None Makes an elevation request along a path, where the elevation data are returned as distance-based samples along that path.
`getElevationForLocations(request:LocationElevationRequest, callback:function(Array<ElevationResult>, ElevationStatus))`	**Return Value:** None Makes an elevation request for a list of discrete locations.

LocationElevationRequest object specification

`google.maps.LocationElevationRequest` object specification

An elevation request sent by the `ElevationService` containing the list of discrete coordinates (`LatLngs`) for which to return elevation data.

Properties	
`locations`	**Type:** `Array<LatLng>` The discrete locations for which to retrieve elevations.

PathElevationRequest object specification

`google.maps.PathElevationRequest` object specification

An elevation query sent by the `ElevationService` containing the path along which to return sampled data. This request defines a continuous path along the earth along which elevation samples should be taken at evenly-spaced distances. All paths from vertex to vertex use segments of the great circle between those two points.

Properties	
`path`	**Type:** `Array<LatLng>` The path along which to collect elevation values.
`samples`	**Type:** `number` Required. The number of equidistant points along the given path for which to retrieve elevation data, including the endpoints. The number of samples must be a value between 2 and 512 inclusive.

ElevationResult object specification

`google.maps.ElevationResult` object specification

The result of an `ElevationService` request, consisting of the set of elevation coordinates and their elevation values. Note that a single request may produce multiple `ElevationResults`.

Properties	
`elevation`	**Type:** `number` The elevation of this point on Earth, in meters above sea level.
`location`	**Type:** `LatLng` The location of this elevation result.
`resolution`	**Type:** `number` The distance, in meters, between sample points from which the elevation was interpolated. This property will be missing if the resolution is not known. Note that elevation data becomes more coarse (larger `resolution` values) when multiple points are passed. To obtain the most accurate elevation value for a point, it should be queried independently.

ElevationStatus class

`google.maps.ElevationStatus` class

The status returned by the `ElevationService` upon completion of an elevation requerst.

Constant	
`INVALID_REQUEST`	This request was invalid.
`OK`	The request did not encounter any errors.
`OVER_QUERY_LIMIT`	The webpage has gone over the requests limit in too short a period of time.
`REQUEST_DENIED`	The webpage is not allowed to use the elevation service for some reason.
`UNKNOWN_ERROR`	A geocoding, directions or elevation request could not be successfully processed, yet the exact reason for the failure is not known.

MaxZoomService class

`google.maps.MaxZoomService` class

A service for obtaining the highest zoom level at which satellite imagery is available for a given location.

Constructor	
`MaxZoomService()`	Creates a new instance of a `MaxZoomService` that can be used to send queries about the maximum zoom level available for satellite imagery.

Methods	
`getMaxZoomAtLatLng(latlng:LatLng|LatLngLiteral, callback:function(MaxZoomResult))`	**Return Value:** None Returns the maximum zoom level available at a particular `LatLng` for the Satellite map type. As this request is asynchronous, you must pass a `callback` function which will be executed upon completion of the request, being passed a `MaxZoomResult`.

MaxZoomResult object specification

`google.maps.MaxZoomResult` object specification

A MaxZoom result in JSON format retrieved from the MaxZoomService.

Properties	
`status`	**Type:** `MaxZoomStatus` Status of the request.
`zoom`	**Type:** `number` The maximum zoom level found at the given `LatLng`.

MaxZoomStatus class

`google.maps.MaxZoomStatus` class

The status returned by the `MaxZoomService` on the completion of a call to `getMaxZoomAtLatLng()`.

Constant	
`ERROR`	There was a problem contacting the Google servers.
`OK`	The response contains a valid `MaxZoomResult`.

DistanceMatrixService class

`google.maps.DistanceMatrixService` class

A service for computing distances between multiple origins and destinations.

Constructor	
`DistanceMatrixService()`	Creates a new instance of a `DistanceMatrixService` that sends distance matrix queries to Google servers.

Methods	
`getDistanceMatrix(request:DistanceMatrixRequest, callback:function(DistanceMatrixResponse, DistanceMatrixStatus))`	**Return Value:** None Issues a distance matrix request.

DistanceMatrixRequest object specification

`google.maps.DistanceMatrixRequest` object specification

A distance matrix query sent by the `DistanceMatrixService` containing arrays of origin and destination locations, and various options for computing metrics.

Properties	
`avoidFerries`	**Type:** `boolean` If true, instructs the Distance Matrix service to avoid ferries where possible. Optional.
`avoidHighways`	**Type:** `boolean` If true, instructs the Distance Matrix service to avoid highways where possible. Optional.
`avoidTolls`	**Type:** `boolean` If true, instructs the Distance Matrix service to avoid toll roads where possible. Optional.
`destinations`	**Type:** `Array<LatLng|string>` An array containing destination address strings and/or `LatLng`s, to which to calculate distance and time. Required.
`durationInTraffic`	**Type:** `boolean` Whether or not we should provide trip durations based on current traffic conditions. Only available to Maps API for Work customers.
`origins`	**Type:** `Array<LatLng|string>` An array containing origin address strings and/or `LatLng`s, from which to calculate distance and time. Required.
`region`	**Type:** `string` Region code used as a bias for geocoding requests. Optional.
`transitOptions`	**Type:** `TransitOptions` Settings that apply only to requests where `travelMode` is TRANSIT. This object will have no effect for other travel modes.

travelMode	**Type:** TravelMode Type of routing requested. Required.
unitSystem	**Type:** UnitSystem Preferred unit system to use when displaying distance. Optional; defaults to metric.

DistanceMatrixResponse object specification

google.maps.DistanceMatrixResponse object specification

The response to a DistanceMatrixService request, consisting of the formatted origin and destination addresses, and a sequence of DistanceMatrixResponseRows, one for each corresponding origin address.

Properties	
destinationAddresses	**Type:** Array<string> The formatted destination addresses.
originAddresses	**Type:** Array<string> The formatted origin addresses.
rows	**Type:** Array<DistanceMatrixResponseRow> The rows of the matrix, corresponding to the origin addresses.

DistanceMatrixResponseRow object specification

google.maps.DistanceMatrixResponseRow object specification

A row of the response to a DistanceMatrixService request, consisting of a sequence of DistanceMatrixResponseElements, one for each corresponding destination address.

Properties	
elements	**Type:** Array<DistanceMatrixResponseElement> The row's elements, corresponding to the destination addresses.

DistanceMatrixResponseElement object specification

`google.maps.DistanceMatrixResponseElement` object specification

A single element of a response to a `DistanceMatrixService` request, which contains the duration and distance from one origin to one destination.

Properties	
`distance`	**Type:** `Distance` The distance for this origin-destination pairing. This property may be undefined as the distance may be unknown.
`duration`	**Type:** `Duration` The duration for this origin-destination pairing. This property may be undefined as the duration may be unknown.
`fare`	**Type:** `TransitFare` The total fare for this origin-destination pairing. Only applicable to transit requests.
`status`	**Type:** `DistanceMatrixElementStatus` The status of this particular origin-destination pairing.

DistanceMatrixStatus class

`google.maps.DistanceMatrixStatus` class

The top-level status about the request in general returned by the `DistanceMatrixService` upon completion of a distance matrix request.

Constant	
`INVALID_REQUEST`	The provided request was invalid.
`MAX_DIMENSIONS_EXCEEDED`	The request contains more than 25 origins, or more than 25 destinations.
`MAX_ELEMENTS_EXCEEDED`	The product of origins and destinations exceeds the per-query limit.

`OK`	The response contains a valid result.
`OVER_QUERY_LIMIT`	Too many elements have been requested within the allowed time period. The request should succeed if you try again after a reasonable amount of time.
`REQUEST_DENIED`	The service denied use of the Distance Matrix service by your web page.
`UNKNOWN_ERROR`	A Distance Matrix request could not be processed due to a server error. The request may succeed if you try again.

DistanceMatrixElementStatus class

`google.maps.DistanceMatrixElementStatus` class

The element-level status about a particular origin-destination pairing returned by the `DistanceMatrixService` upon completion of a distance matrix request.

Constant	
`NOT_FOUND`	The origin and/or destination of this pairing could not be geocoded.
`OK`	The response contains a valid result.
`ZERO_RESULTS`	No route could be found between the origin and destination.

Attribution object specification

`google.maps.Attribution` object specification

Properties	
`iosDeepLinkId`	**Type:** `string` The iOS deep link to associate with this place when a user saves the place. When the user views the place in an iOS app, this URL will serve as the link on the `source` string. If there is no deep link or the app that handles the deep link is not present, the webURL will be used instead.
`source`	**Type:** `string` The source (origin) to associate with this place when it is saved by a user. For example, this could be the name of your website or application. The user who saved the place will see this source when they view the place in Google Maps. source is required for an

	see this source when they view the place in Google Maps. `source` is required for an `Attribution` to be considered valid. If it is not provided an error will be thrown.
`webUrl`	**Type:** `string` The URL (http or https) of the page to associate with this place when a user saves the place. When the user views the place in a desktop or Android app, this URL will serve as the link on the `source` string. When the user views the place in an iOS app, and there is no deep link provided or the app that handles the deep link is not present, this URL will serve as the link on the `source` string.

MarkerPlace object specification

`google.maps.MarkerPlace` object specification

Properties	
`location`	**Type:** `LatLng`|`LatLngLiteral` The `LatLng` of the entity described by this Place. This must be provided for the `Place` to be considered valid.
`placeId`	**Type:** `string` The place ID of the place (such as a business or point of interest). The place ID is a unique identifier of a place in the Google Maps database. Note that the `placeId` is the most accurate way of identifying a place. If possible, you should specify the `placeId` rather than a `placeQuery`. A place ID can be retrieved from any request to the Places API, such as a TextSearch. Place IDs can also be retrieved from requests to the Geocoding API.
`query`	**Type:** `string` A search query describing the place (such as a business or point of interest). An example query would be "Quay, Upper Level, Overseas Passenger Terminal 5 Hickson Road, The Rocks NSW". If possible, you should specify the `placeId` rather than a `placeQuery`. The API does not guarantee the accuracy of resolving the query string to a place. If both the `placeId` and `placeQuery` are provided, an error is thrown.

SaveWidget class

`google.maps.SaveWidget` class

A control that users can use to save a place to Google Maps from your website. In this context, 'place' means a business, point of interest or geographic location. The `SaveWidget` has a fixed height of 22px.

The `SaveWidget` is only available when `signed-in=true` has been passed as a URL parameter in the bootstrap request.

Constructor	
`SaveWidget(container:Node, opts?:SaveWidgetOptions)`	Creates a new `SaveWidget`, and renders it in a given div.

Methods	
`getAttribution()`	**Return Value:** `Attribution` Returns the `Attribution` associated with this `SaveWidget`.
`getPlace()`	**Return Value:** `MarkerPlace` Returns the `Place` associated with this `SaveWidget`.
`setAttribution(attribution:Attribution)`	**Return Value:** None Sets the `Attribution` associated with this `SaveWidget`.
`setOptions(opts:SaveWidgetOptions)`	**Return Value:** None
`setPlace(place:MarkerPlace)`	**Return Value:** None Changes the `Place` associated with this `SaveWidget`.

SaveWidgetOptions object specification

`google.maps.SaveWidgetOptions` object specification

Properties	
`attribution`	**Type:** `Attribution` Contains all the information needed to identify your application as the source of a save to Google Maps. The `SaveWidget` will show this information when a user saves the associated `Place`.

place	**Type:** MarkerPlace Describes a Place that a user can save from your application to their personalized map. In this context, 'place' means a business, point of interest or geographic location.

MapType object specification

`google.maps.MapType` object specification

This interface defines the map type, and is typically used for custom map types. Immutable.

Methods	
`getTile(tileCoord:Point, zoom:number, ownerDocument:Document)`	**Return Value:** Node Returns a tile for the given tile coordinate (x, y) and zoom level. This tile will be appended to the given ownerDocument. Not available for base map types.
`releaseTile(tile:Node)`	**Return Value:** None Releases the given tile, performing any necessary cleanup. The provided tile will have already been removed from the document. Optional.

Properties	
`alt`	**Type:** `string` Alt text to display when this MapType's button is hovered over in the MapTypeControl. Optional.
`maxZoom`	**Type:** `number` The maximum zoom level for the map when displaying this MapType. Required for base MapTypes, ignored for overlay MapTypes.
`minZoom`	**Type:** `number` The minimum zoom level for the map when displaying this MapType. Optional; defaults to 0.
`name`	**Type:** `string` Name to display in the MapTypeControl. Optional.

`projection`	**Type:** `Projection` The Projection used to render this MapType. Optional; defaults to Mercator.
`radius`	**Type:** `number` Radius of the planet for the map, in meters. Optional; defaults to Earth's equatorial radius of 6378137 meters.
`tileSize`	**Type:** `Size` The dimensions of each tile. Required.

MapTypeRegistry class

`google.maps.MapTypeRegistry` class

This class extends `MVCObject`.

Constructor	
`MapTypeRegistry()`	The `MapTypeRegistry` holds the collection of custom map types available to the map for its use. The API consults this registry when providing the list of avaiable map types within controls, for example.

Methods	
`set(id:string, mapType:MapType|undefined)`	**Return Value:** None Sets the registry to associate the passed string identifier with the passed MapType.

Projection object specification

`google.maps.Projection` object specification

Methods	
`fromLatLngToPoint(`	**Return Value:** `Point`

latLng:LatLng, point?:Point)	Translates from the LatLng cylinder to the Point plane. This interface specifies a function which implements translation from given `LatLng` values to world coordinates on the map projection. The Maps API calls this method when it needs to plot locations on screen. `Projection` objects must implement this method.
fromPointToLatLng(pixel:Point, nowrap?:boolean)	**Return Value:** `LatLng` This interface specifies a function which implements translation from world coordinates on a map projection to `LatLng` values. The Maps API calls this method when it needs to translate actions on screen to positions on the map. `Projection` objects must implement this method.

`google.maps.`

ImageMapTypeOptions object specification

`google.maps.ImageMapTypeOptions` object specification

This class is used to create a MapType that renders image tiles.

Properties	
alt	**Type:** `string` Alt text to display when this MapType's button is hovered over in the MapTypeControl.
getTileUrl	**Type:** `function(Point, number): string` Returns a string (URL) for given tile coordinate (x, y) and zoom level.
maxZoom	**Type:** `number` The maximum zoom level for the map when displaying this MapType.
minZoom	**Type:** `number` The minimum zoom level for the map when displaying this MapType. Optional.
name	**Type:** `string` Name to display in the MapTypeControl.
opacity	**Type:** `number` The opacity to apply to the tiles. The opacity should be specified as a float value between 0

	and 1.0, where 0 is fully transparent and 1 is fully opaque.
`tileSize`	**Type:** `Size` The tile size.

StyledMapType class

`google.maps.StyledMapType` class

Creates a `MapType` with a custom style.

This class extends `MVCObject`.

Constructor	
`StyledMapType(styles:Array<MapTypeStyle>, options?:StyledMapTypeOptions)`	Creates a styled `MapType` with the specified options. The `StyledMapType` takes an array of `MapTypeStyle`s, where each `MapTypeStyle` is applied to the map consecutively. A later `MapTypeStyle` that applies the same `MapTypeStyler`s to the same selectors as an earlier `MapTypeStyle` will override the earlier `MapTypeStyle`.

Methods	
`getTile(tileCoord:Point, zoom:number, ownerDocument:Document)`	**Return Value:** `Node`
`releaseTile(tile:Node)`	**Return Value:** `None`

Properties	
`alt`	**Type:** `string`
`maxZoom`	**Type:** `number`
`minZoom`	**Type:** `number`
`name`	**Type:** `string`

`projection`	**Type:** `Projection`
`radius`	**Type:** `number`
`tileSize`	**Type:** `Size`

StyledMapTypeOptions object specification

`google.maps.StyledMapTypeOptions` object specification

This class is used to specify options when creating a `StyledMapType`. These options cannot be changed after the `StyledMapType` is instantiated.

Properties	
`alt`	**Type:** `string` Text to display when this `MapType`'s button is hovered over in the map type control.
`maxZoom`	**Type:** `number` The maximum zoom level for the map when displaying this `MapType`. Optional.
`minZoom`	**Type:** `number` The minimum zoom level for the map when displaying this `MapType`. Optional.
`name`	**Type:** `string` The name to display in the map type control.

MapTypeStyle object specification

`google.maps.MapTypeStyle` object specification

The `MapTypeStyle` is a collection of selectors and stylers that define how the map should be styled. Selectors specify what map elements should be affected and stylers specify how those elements should be modified.

Properties	
`elementType`	**Type:** `MapTypeStyleElementType` Selects the element type to which a styler should be applied. An element type distinguishes between the different representations of a feature. Optional; if `elementType` is not specified, the value is assumed to be `'all'`.
`featureType`	**Type:** `MapTypeStyleFeatureType` Selects the feature, or group of features, to which a styler should be applied. Optional; if `featureType` is not specified, the value is assumed to be `'all'`.
`stylers`	**Type:** `Array<MapTypeStyler>` The style rules to apply to the selectors. The rules are applied to the map's elements in the order they are listed in this array.

MapTypeStyleFeatureType object specification

`google.maps.MapTypeStyleFeatureType` object specification

Possible values for feature types. Specify these values as strings, i.e. `'administrative'` or `'poi.park'`. Stylers applied to a parent feature type automatically apply to all child feature types. Note however that parent features may include some additional features that are not included in one of their child feature types.

Constant	
`administrative`	Apply the rule to administrative areas.
`administrative.country`	Apply the rule to countries.
`administrative.land_parcel`	Apply the rule to land parcels.
`administrative.locality`	Apply the rule to localities.
`administrative.neighborhood`	Apply the rule to neighborhoods.
`administrative.province`	Apply the rule to provinces.
`all`	Apply the rule to all selector types.
`landscape`	Apply the rule to landscapes.

`landscape`	Apply the rule to landscapes.
`landscape.man_made`	Apply the rule to man made structures.
`landscape.natural`	Apply the rule to natural features.
`landscape.natural.landcover`	Apply the rule to landcover.
`landscape.natural.terrain`	Apply the rule to terrain.
`poi`	Apply the rule to points of interest.
`poi.attraction`	Apply the rule to attractions for tourists.
`poi.business`	Apply the rule to businesses.
`poi.government`	Apply the rule to government buildings.
`poi.medical`	Apply the rule to emergency services (hospitals, pharmacies, police, doctors, etc).
`poi.park`	Apply the rule to parks.
`poi.place_of_worship`	Apply the rule to places of worship, such as churches, temples, or mosques.
`poi.school`	Apply the rule to schools.
`poi.sports_complex`	Apply the rule to sports complexes.
`road`	Apply the rule to all roads.
`road.arterial`	Apply the rule to arterial roads.
`road.highway`	Apply the rule to highways.
`road.highway.controlled_access`	Apply the rule to controlled-access highways.
`road.local`	Apply the rule to local roads.
`transit`	Apply the rule to all transit stations and lines.
`transit.line`	Apply the rule to transit lines.
`transit.station`	Apply the rule to all transit stations.
`transit.station.airport`	Apply the rule to airports.
`transit.station.bus`	Apply the rule to bus stops.
`transit.station.rail`	Apply the rule to rail stations.
`water`	Apply the rule to bodies of water.

`water`	Apply the rule to bodies of water.

MapTypeStyleElementType object specification

`google.maps.MapTypeStyleElementType` object specification

Each `MapTypeStyleElementType` distinguishes between the different representations of a feature.

Constant	
`all`	Apply the rule to all elements of the specified feature.
`geometry`	Apply the rule to the feature's geometry.
`geometry.fill`	Apply the rule to the fill of the feature's geometry.
`geometry.stroke`	Apply the rule to the stroke of the feature's geometry.
`labels`	Apply the rule to the feature's labels.
`labels.icon`	Apply the rule to icons within the feature's labels.
`labels.text`	Apply the rule to the text in the feature's label.
`labels.text.fill`	Apply the rule to the fill of the text in the feature's labels.
`labels.text.stroke`	Apply the rule to the stroke of the text in the feature's labels.

MapTypeStyler object specification

`google.maps.MapTypeStyler` object specification

A styler affects how a map's elements will be styled. Each `MapTypeStyler` should contain one and only one key. If more than one key is specified in a single `MapTypeStyler`, all but one will be ignored. For example: `var rule = {hue: '#ff0000'}`.

Properties	
`color`	**Type:** `string` Sets the color of the feature. Valid values: An RGB hex string, i.e. `'#ff0000'`.

`gamma`	**Type:** `number` Modifies the gamma by raising the lightness to the given power. Valid values: Floating point numbers, [0.01, 10], with 1.0 representing no change.
`hue`	**Type:** `string` Sets the hue of the feature to match the hue of the color supplied. Note that the saturation and lightness of the feature is conserved, which means that the feature will not match the color supplied exactly. Valid values: An RGB hex string, i.e. `'#ff0000'`.
`invert_lightness`	**Type:** `boolean` A value of `true` will invert the lightness of the feature while preserving the hue and saturation.
`lightness`	**Type:** `number` Shifts lightness of colors by a percentage of the original value if decreasing and a percentage of the remaining value if increasing. Valid values: [-100, 100].
`saturation`	**Type:** `number` Shifts the saturation of colors by a percentage of the original value if decreasing and a percentage of the remaining value if increasing. Valid values: [-100, 100].
`visibility`	**Type:** `string` Sets the visibility of the feature. Valid values: 'on', 'off' or 'simplified'.
`weight`	**Type:** `number` Sets the weight of the feature, in pixels. Valid values: Integers greater than or equal to zero.

BicyclingLayer class

`google.maps.BicyclingLayer` class

A layer showing bike lanes and paths.

This class extends `MVCObject`.

Constructor	
`BicyclingLayer()`	A layer that displays bike lanes and paths and demotes large roads.

Methods	
`getMap()`	**Return Value:** Map Returns the map on which this layer is displayed.
`setMap(map:Map)`	**Return Value:** None Renders the layer on the specified map. If map is set to null, the layer will be removed.

FusionTablesLayer class

`google.maps.FusionTablesLayer` class

A `FusionTablesLayer` allows you to display data from a Google Fusion Table on a map, as a rendered layer. (See https://developers.google.com/fusiontables/ for more information about Fusion Tables).

This class extends `MVCObject`.

Constructor	
`FusionTablesLayer( options:`*`FusionTablesLayerOptions`*`)`	A layer that displays data from a Fusion Table.

Methods	
`getMap()`	**Return Value:** Map Returns the map on which this layer is displayed.
`setMap(map:Map)`	**Return Value:** None Renders the layer on the specified map. If map is set to null, the layer will be removed.
`setOptions( options:`*`FusionTablesLayerOptions`*`)`	**Return Value:** None

Events

`click`	**Arguments:** *FusionTablesMouseEvent* This event is fired when a feature in the layer is clicked.

FusionTablesLayerOptions object specification

`google.maps.FusionTablesLayerOptions` object specification

This object defines the properties that can be set on a `FusionTablesLayer` object.

Properties	
`clickable`	**Type:** `boolean` If true, the layer receives mouse events. Default value is true.
`heatmap`	**Type:** `FusionTablesHeatmap` Options which define the appearance of the layer as a heatmap.
`map`	**Type:** `Map` The map on which to display the layer.
`query`	**Type:** `FusionTablesQuery` Options defining the data to display.
`styles`	**Type:** `Array<FusionTablesStyle>` An array of up to 5 style specifications, which control the appearance of features within the layer.
`suppressInfoWindows`	**Type:** `boolean` Suppress the rendering of info windows when layer features are clicked.

FusionTablesQuery object specification

`google.maps.FusionTablesQuery` object specification

Specifies the data to retrieve from a Fusion Tables.

Properties	
`from`	**Type:** `string` The ID of the Fusion Tables table to display. This ID can be found in the table's URL, as the value of the `dsrcid` parameter. Required.
`limit`	**Type:** `number` Limit on the number of results returned by the query.
`offset`	**Type:** `number` Offset into the sorted results.
`orderBy`	**Type:** `string` The method by which to sort the results. Accepts either of: • A column name. The column name may be suffixed with ASC or DESC (e.g. `col2 DESC`) to specify ascending or descending sort. • An ST_DISTANCE spatial relationship (sort by distance). A column and the coordinate from which to calculate distance must be passed, for example, `orderBy: 'ST_DISTANCE(col1, LATLNG(1.2, 3.4))'`.
`select`	**Type:** `string` A column, containing geographic features to be displayed on the map. See Fusion Tables Setup in the Maps API documentation for information about valid columns.
`where`	**Type:** `string` The SQL predicate to be applied to the layer.

FusionTablesStyle object specification

`google.maps.FusionTablesStyle` object specification

Controls the appearance of a set of features within a `FusionTablesLayer`. Features which match the provided SQL predicate will be styled with the supplied options.

Properties	
`markerOptions`	**Type:** `FusionTablesMarkerOptions` Options which control the appearance of point features.

`polygonOptions`	**Type:** `FusionTablesPolygonOptions` Options which control the appearance of polygons.
`polylineOptions`	**Type:** `FusionTablesPolylineOptions` Options which control the appearance of polylines.
`where`	**Type:** `string` The SQL predicate to be applied to the layer.

FusionTablesHeatmap object specification

`google.maps.FusionTablesHeatmap` object specification

Specifies the appearance for a FusionTablesLayer when rendered as a heatmap.

Properties	
`enabled`	**Type:** `boolean` If true, render the layer as a heatmap.

FusionTablesMarkerOptions object specification

`google.maps.FusionTablesMarkerOptions` object specification

Options which control the appearance of point features in a `FusionTablesLayer`.

Properties	
`iconName`	**Type:** `string` The name of a Fusion Tables supported icon

FusionTablesPolygonOptions object specification

`google.maps.FusionTablesPolygonOptions` object specification

Options which control the appearance of polygons in a `FusionTablesLayer`.

Properties	
`fillColor`	**Type:** `string` The fill color, defined by a six-digit hexadecimal number in RRGGBB format (e.g. `#00AAFF`).
`fillOpacity`	**Type:** `number` The fill opacity between 0.0 and 1.0.
`strokeColor`	**Type:** `string` The fill color, defined by a six-digit hexadecimal number in RRGGBB format (e.g. `#00AAFF`).
`strokeOpacity`	**Type:** `number` The stroke opacity between 0.0 and 1.0.
`strokeWeight`	**Type:** `number` The stroke width in pixels, between 0 and 10.

FusionTablesPolylineOptions object specification

`google.maps.FusionTablesPolylineOptions` object specification

Options which control the appearance of polylines in a `FusionTablesLayer`.

Properties	
`strokeColor`	**Type:** `string` The fill color, defined by a six-digit hexadecimal number in RRGGBB format (e.g. `#00AAFF`).
`strokeOpacity`	**Type:** `number` The stroke opacity between 0.0 and 1.0.
`strokeWeight`	**Type:** `number` The stroke width in pixels.

FusionTablesMouseEvent object specification

`google.maps.FusionTablesMouseEvent` object specification

The properties of a mouse event on a `FusionTablesLayer`.

Properties	
`infoWindowHtml`	**Type:** `string` Pre-rendered HTML content, as placed in the infowindow by the default UI.
`latLng`	**Type:** `LatLng` The position at which to anchor an infowindow on the clicked feature.
`pixelOffset`	**Type:** `Size` The offset to apply to an infowindow anchored on the clicked feature.
`row`	**Type:** `Object<FusionTablesCell>` A collection of `FusionTablesCell` objects, indexed by column name, representing the contents of the table row which included the clicked feature.

FusionTablesCell object specification

`google.maps.FusionTablesCell` object specification

Describes a single cell from a Fusion Tables table.

Properties	
`columnName`	**Type:** `string` The name of the column in which the cell was located.
`value`	**Type:** `string` The contents of the cell.

KmlLayer class

`google.maps.KmlLayer` class

A `KmlLayer` adds geographic markup to the map from a KML, KMZ or GeoRSS file that is hosted on a publicly accessible web server. A `KmlFeatureData` object is provided for each feature when clicked.

This class extends `MVCObject`.

Constructor	
`KmlLayer( opts?:KmlLayerOptions)`	Creates a `KmlLayer` which renders the contents of the specified KML/KMZ file (https://developers.google.com/kml/documentation/kmlreference) or GeoRSS file (http://www.georss.org).

Methods	
`getDefaultViewport()`	**Return Value:** `LatLngBounds` Get the default viewport for the layer being displayed.
`getMap()`	**Return Value:** `Map` Get the map on which the KML Layer is being rendered.
`getMetadata()`	**Return Value:** `KmlLayerMetadata` Get the metadata associated with this layer, as specified in the layer markup.
`getStatus()`	**Return Value:** `KmlLayerStatus` Get the status of the layer, set once the requested document has loaded.
`getUrl()`	**Return Value:** `string` Gets the URL of the KML file being displayed.
`getZIndex()`	**Return Value:** `number` Gets the z-index of the KML Layer.
`setMap(map:Map)`	**Return Value:** None Renders the KML Layer on the specified map. If map is set to null, the layer is removed.
`setUrl(url:string)`	**Return Value:** None

	Sets the URL of the KML file to display.

`setZIndex(`
`zIndex:number)`
	Return Value: None
	Sets the z-index of the KML Layer.

Events	
`click`	**Arguments:** *`KmlMouseEvent`* This event is fired when a feature in the layer is clicked.
`defaultviewport_changed`	**Arguments:** None This event is fired when the KML layers default viewport has changed.
`status_changed`	**Arguments:** None This event is fired when the KML layer has finished loading. At this point it is safe to read the status property to determine if the layer loaded successfully.

KmlLayerOptions object specification

`google.maps.KmlLayerOptions` object specification

This object defines the properties that can be set on a `KmlLayer` object.

Properties	
`clickable`	**Type:** `boolean` If true, the layer receives mouse events. Default value is true.
`map`	**Type:** `Map` The map on which to display the layer.
`preserveViewport`	**Type:** `boolean` By default, the input map is centered and zoomed to the bounding box of the contents of the layer. If this option is set to `true`, the viewport is left unchanged, unless the map's center and zoom were never set.
`screenOverlays`	**Type:** `boolean`

	Whether to render the screen overlays. Default true.
`suppressInfoWindows`	**Type:** `boolean` Suppress the rendering of info windows when layer features are clicked.
`url`	**Type:** `string` The URL of the KML document to display.
`zIndex`	**Type:** `number` The z-index of the layer.

KmlLayerMetadata object specification

`google.maps.KmlLayerMetadata` object specification

Metadata for a single KML layer, in JSON format.

Properties	
`author`	**Type:** `KmlAuthor` The layer's `<atom:author>`, extracted from the layer markup.
`description`	**Type:** `string` The layer's `<description>`, extracted from the layer markup.
`hasScreenOverlays`	**Type:** `boolean` Whether the layer has any screen overlays.
`name`	**Type:** `string` The layer's `<name>`, extracted from the layer markup.
`snippet`	**Type:** `string` The layer's `<Snippet>`, extracted from the layer markup

KmlLayerStatus class

`google.maps.KmlLayerStatus` class

The status returned by `KmlLayer` on the completion of loading a document.

Constant	
`DOCUMENT_NOT_FOUND`	The document could not be found. Most likely it is an invalid URL, or the document is not publicly available.
`DOCUMENT_TOO_LARGE`	The document exceeds the file size limits of KmlLayer.
`FETCH_ERROR`	The document could not be fetched.
`INVALID_DOCUMENT`	The document is not a valid KML, KMZ or GeoRSS document.
`INVALID_REQUEST`	The `KmlLayer` is invalid.
`LIMITS_EXCEEDED`	The document exceeds the feature limits of KmlLayer.
`OK`	The layer loaded successfully.
`TIMED_OUT`	The document could not be loaded within a reasonable amount of time.
`UNKNOWN`	The document failed to load for an unknown reason.

KmlMouseEvent object specification

`google.maps.KmlMouseEvent` object specification

The properties of a click event on a KML/KMZ or GeoRSS document.

Properties	
`featureData`	**Type:** `KmlFeatureData` A `KmlFeatureData` object, containing information about the clicked feature.
`latLng`	**Type:** `LatLng` The position at which to anchor an infowindow on the clicked feature.
`pixelOffset`	**Type:** `Size` The offset to apply to an infowindow anchored on the clicked feature.

KmlFeatureData object specification

`google.maps.KmlFeatureData` object specification

Data for a single KML feature in JSON format, returned when a KML feature is clicked. The data contained in this object mirrors that associated with the feature in the KML or GeoRSS markup in which it is declared.

Properties	
`author`	**Type:** `KmlAuthor` The feature's `<atom:author>`, extracted from the layer markup (if specified).
`description`	**Type:** `string` The feature's `<description>`, extracted from the layer markup.
`id`	**Type:** `string` The feature's `<id>`, extracted from the layer markup. If no `<id>` has been specified, a unique ID will be generated for this feature.
`infoWindowHtml`	**Type:** `string` The feature's balloon styled text, if set.
`name`	**Type:** `string` The feature's `<name>`, extracted from the layer markup.
`snippet`	**Type:** `string` The feature's `<Snippet>`, extracted from the layer markup.

KmlAuthor object specification

`google.maps.KmlAuthor` object specification

Contains details of the author of a KML document or feature.

Properties	
`email`	**Type:** `string` The author's e-mail address, or an empty string if not specified.

`name`	**Type:** `string` The author's name, or an empty string if not specified.
`uri`	**Type:** `string` The author's home page, or an empty string if not specified.

TrafficLayer class

`google.maps.TrafficLayer` class

A traffic layer.

This class extends `MVCObject`.

Constructor	
`TrafficLayer()`	A layer that displays current road traffic.

Methods	
`getMap()`	**Return Value:** `Map` Returns the map on which this layer is displayed.
`setMap(map:Map)`	**Return Value:** None Renders the layer on the specified map. If map is set to null, the layer will be removed.

TransitLayer class

`google.maps.TransitLayer` class

A transit layer.

This class extends `MVCObject`.

Constructor

`TransitLayer()`	A layer that displays transit lines.

Methods	
`getMap()`	**Return Value:** `Map` Returns the map on which this layer is displayed.
`setMap(map:Map)`	**Return Value:** None Renders the layer on the specified map. If map is set to null, the layer will be removed.

StreetViewPanorama class

`google.maps.StreetViewPanorama` class

Displays the panorama for a given `LatLng` or panorama ID. A `StreetViewPanorama` object provides a Street View "viewer" which can be stand-alone within a separate `<div>` or bound to a `Map`.

This class extends `MVCObject`.

Constructor	
`StreetViewPanorama(container:Node, opts?:StreetViewPanoramaOptions)`	Creates a panorama with the passed `StreetViewPanoramaOptions`.

Methods	
`getLinks()`	**Return Value:** `Array<StreetViewLink>` Returns the set of navigation links for the Street View panorama.
`getLocation()`	**Return Value:** `StreetViewLocation` Returns the StreetViewLocation of the current panorama.
`getPano()`	**Return Value:** `string` Returns the current panorama ID for the Street View panorama. This id is stable within the browser's current session only.

	session only.
`getPhotographerPov()`	**Return Value:** `StreetViewPov` Returns the heading and pitch of the photographer when this panorama was taken. For Street View panoramas on the road, this also reveals in which direction the car was travelling. This data is available after the `pano_changed` event.
`getPosition()`	**Return Value:** `LatLng` Returns the current `LatLng` position for the Street View panorama.
`getPov()`	**Return Value:** `StreetViewPov` Returns the current point of view for the Street View panorama.
`getStatus()`	**Return Value:** `StreetViewStatus` Returns the status of the panorama on completion of the `setPosition()` or `setPano()` request.
`getVisible()`	**Return Value:** `boolean` Returns `true` if the panorama is visible. It does not specify whether Street View imagery is available at the specified position.
`getZoom()`	**Return Value:** `number` Returns the zoom level of the panorama. Fully zoomed-out is level 0, where the field of view is 180 degrees. Zooming in increases the zoom level.
`registerPanoProvider( provider:function( string):StreetViewPanoramaData)`	**Return Value:** None Set the custom panorama provider called on pano change to load custom panoramas.
`setLinks( links:Array<StreetViewLink>)`	**Return Value:** None Sets the set of navigation links for the Street View panorama.
`setOptions( options:`*`StreetViewPanoramaOptions`*`)`	**Return Value:** None Sets a collection of key-value pairs.
`setPano(pano:string)`	**Return Value:** None Sets the current panorama ID for the Street View panorama.

	Sets the current panorama ID for the Street View panorama.
`setPosition(latLng:LatLng|LatLngLiteral)`	**Return Value:** None Sets the current `LatLng` position for the Street View panorama.
`setPov(pov:`*`StreetViewPov`*`)`	**Return Value:** None Sets the point of view for the Street View panorama.
`setVisible(flag:boolean)`	**Return Value:** None Sets to `true` to make the panorama visible. If set to `false`, the panorama will be hidden whether it is embedded in the map or in its own `<div>`.
`setZoom(zoom:number)`	**Return Value:** None Sets the zoom level of the panorama. Fully zoomed-out is level 0, where the field of view is 180 degrees. Zooming in increases the zoom level.

Properties

`controls`	**Type:** `Array<MVCArray<Node>>` Additional controls to attach to the panorama. To add a control to the panorama, add the control's `<div>` to the `MVCArray` corresponding to the `ControlPosition` where it should be rendered.

Events

`clicktogo_changed`	**Arguments:** None This event is fired when the panorama's clickToGo is enabled or disabled.
`closeclick`	**Arguments:** `Event` This event is fired when the close button is clicked.
`links_changed`	**Arguments:** None This event is fired when the panorama's links change. The links change asynchronously following a pano id change.
`pano_changed`	**Arguments:** None This event is fired when the panorama's pano id changes. The pano may change as the user navigates through the panorama or the position is manually set.

	Note that not all position changes trigger a pano_changed.
`position_changed`	**Arguments:** None This event is fired when the panorama's position changes. The position changes as the user navigates through the panorama or the position is set manually.
`pov_changed`	**Arguments:** None This event is fired when the panorama's point-of-view changes. The point of view changes as the pitch, zoom, or heading changes.
`resize`	**Arguments:** None Developers should trigger this event on the panorama when its div changes size: `google.maps.event.trigger(panorama, 'resize')`.
`scrollwheel_changed`	**Arguments:** None This event is fired when the panorama's scrollWheel is enabled or disabled.
`status_changed`	**Arguments:** None This event is fired after every panorama lookup by id or location, via `setPosition()` or `setPano()`.
`visible_changed`	**Arguments:** None This event is fired when the panorama's visibility changes. The visibility is changed when the Pegman is dragged onto the map, the close button is clicked, or `setVisible()` is called.
`zoom_changed`	**Arguments:** None This event is fired when the panorama's zoom level changes.

StreetViewPanoramaOptions object specification

`google.maps.StreetViewPanoramaOptions` object specification

Options defining the properties of a `StreetViewPanorama` object.

Properties	
`addressControl`	**Type:** `boolean` The enabled/disabled state of the address control.
`addressControlOptions`	**Type:** `StreetViewAddressControlOptions`

`addressControlOptions`	**Type:** `StreetViewAddressControlOptions` The display options for the address control.
`clickToGo`	**Type:** `boolean` The enabled/disabled state of click-to-go.
`disableDefaultUI`	**Type:** `boolean` Enables/disables all default UI. May be overridden individually.
`disableDoubleClickZoom`	**Type:** `boolean` Enables/disables zoom on double click. Disabled by default.
`enableCloseButton`	**Type:** `boolean` If `true`, the close button is displayed. Disabled by default.
`imageDateControl`	**Type:** `boolean` The enabled/disabled state of the imagery acquisition date control. Disabled by default.
`linksControl`	**Type:** `boolean` The enabled/disabled state of the links control.
`panControl`	**Type:** `boolean` The enabled/disabled state of the pan control.
`panControlOptions`	**Type:** `PanControlOptions` The display options for the pan control.
`pano`	**Type:** `string` The panorama ID, which should be set when specifying a custom panorama.
`panoProvider`	**Type:** `function(string): StreetViewPanoramaData` Custom panorama provider, which takes a string pano id and returns an object defining the panorama given that id. This function must be defined to specify custom panorama imagery.
`position`	**Type:** `LatLng|LatLngLiteral` The `LatLng` position of the Street View panorama.
`pov`	**Type:** `StreetViewPov` The camera orientation, specified as heading and pitch, for the panorama.
`scrollwheel`	**Type:** `boolean`

	If false, disables scrollwheel zooming in Street View. The scrollwheel is enabled by default.
`visible`	**Type:** `boolean` If `true`, the Street View panorama is visible on load.
`zoomControl`	**Type:** `boolean` The enabled/disabled state of the zoom control.
`zoomControlOptions`	**Type:** `ZoomControlOptions` The display options for the zoom control.

StreetViewAddressControlOptions object specification

`google.maps.StreetViewAddressControlOptions` object specification

Options for the rendering of the Street View address control.

Properties	
`position`	**Type:** `ControlPosition` Position id. This id is used to specify the position of the control on the map. The default position is TOP_LEFT.

StreetViewLink object specification

`google.maps.StreetViewLink` object specification

A collection of references to adjacent Street View panos.

Properties	
`description`	**Type:** `string` A localized string describing the link.
`heading`	**Type:** `number` The heading of the link.

`pano`	**Type:** `string` A unique identifier for the panorama. This id is stable within a session but unstable across sessions.

StreetViewPov object specification

`google.maps.StreetViewPov` object specification

A point of view object which specifies the camera's orientation at the Street View panorama's position. The point of view is defined as heading and pitch.

Properties	
`heading`	**Type:** `number` The camera heading in degrees relative to true north. True north is 0°, east is 90°, south is 180°, west is 270°.
`pitch`	**Type:** `number` The camera pitch in degrees, relative to the street view vehicle. Ranges from 90° (directly upwards) to -90° (directly downwards).

StreetViewLocationRequest object specification

`google.maps.StreetViewLocationRequest` object specification

A Street View request to be sent with `getPanorama`. `StreetViewLocationRequest` lets you search for a Street View panorama at a specified location.

Properties	
`location`	**Type:** `LatLng`|`LatLngLiteral` Specifies the location where to search for a Street View panorama.
`preference`	**Type:** `StreetViewPreference` Sets a preference for which panorama should be found within the radius: the one nearest to the provided location, or the best one within the radius.

`radius`	**Type:** `number` Sets a radius in meters in which to search for a panorama. Defaults to 50 when not supplied.
`source`	**Type:** `StreetViewSource` Specifies the source of panoramas to search. This allows a restriction to search for just outdoor panoramas for example. If not specified it is set to `DEFAULT`.

StreetViewPanoRequest object specification

`google.maps.StreetViewPanoRequest` object specification

A `StreetViewPanoRequest` is used with the `getPanorama` to find a panorama with a specified ID.

Properties	
`pano`	**Type:** `string` Specifies the pano ID to search for.

StreetViewPanoramaData object specification

`google.maps.StreetViewPanoramaData` object specification

The representation of a panorama returned from the provider defined using `registerPanoProvider`.

Properties	
`copyright`	**Type:** `string` Specifies the copyright text for this panorama.
`imageDate`	**Type:** `string` Specifies the year and month in which the imagery in this panorama was acquired. The date string is in the form YYYY-MM.
`links`	**Type:** `Array<StreetViewLink>`

	Specifies the navigational links to adjacent panoramas.
`location`	**Type:** `StreetViewLocation` Specifies the location meta-data for this panorama.
`tiles`	**Type:** `StreetViewTileData` Specifies the custom tiles for this panorama.

StreetViewLocation object specification

`google.maps.StreetViewLocation` object specification

A representation of a location in the Street View panorama.

Properties	
`description`	**Type:** `string` A localized string describing the location.
`latLng`	**Type:** `LatLng` The latlng of the panorama.
`pano`	**Type:** `string` A unique identifier for the panorama. This is stable within a session but unstable across sessions.
`shortDescription`	**Type:** `string` Short description of the location.

StreetViewPreference class

`google.maps.StreetViewPreference` class

Options that bias a search result towards returning a Street View panorama that is nearest to the request location, or a panorama that is considered most likely to be what the user wants to see.

Constant	
`BEST`	Return the Street View panorama that is considered most likely to be what the user wants to see. The best result is determined by algorithms based on user research and parameters such as recognised points of interest, image quality, and distance from the given location.
`NEAREST`	Return the Street View panorama that is the shortest distance from the provided location. This works well only within a limited radius. The recommended radius is 1km or less.

StreetViewSource class

`google.maps.StreetViewSource` class

Identifiers to limit Street View searches to selected sources.

Constant	
`DEFAULT`	Uses the default sources of Street View, searches will not be limited to specific sources.
`OUTDOOR`	Limits Street View searches to outdoor collections only.

StreetViewTileData object specification

`google.maps.StreetViewTileData` object specification

The properties of the tile set used in a Street View panorama.

Methods	
`getTileUrl(pano:string, tileZoom:number, tileX:number, tileY:number)`	**Return Value:** `string` Gets the tile image URL for the specified tile. pano is the panorama ID of the Street View tile. `tileZoom` is the zoom level of the tile. `tileX` is the x-coordinate of the tile. `tileY` is the y-coordinate of the tile. Returns the URL for the tile image

	Returns the URL for the tile image.

Properties	
`centerHeading`	**Type:** `number` The heading (in degrees) at the center of the panoramic tiles.
`tileSize`	**Type:** `Size` The size (in pixels) at which tiles will be rendered.
`worldSize`	**Type:** `Size` The size (in pixels) of the whole panorama's "world".

StreetViewService class

`google.maps.StreetViewService` class

A `StreetViewService` object performs searches for Street View data.

Methods	
`getPanorama(request:StreetViewLocationRequest|StreetViewPanoRequest, callback:function(StreetViewPanoramaData, StreetViewStatus))`	**Return Value:** None Retrieves the `StreetViewPanoramaData` for a panorama that matches the supplied Street View query request. The `StreetViewPanoramaData` is passed to the provided callback.

StreetViewStatus class

`google.maps.StreetViewStatus` class

The status returned by the `StreetViewService` on completion of a Street View request.

Constant	
`OK`	The request was successful.

UNKNOWN_ERROR	The request could not be successfully processed, yet the exact reason for failure is unknown.
ZERO_RESULTS	There are no nearby panoramas.

StreetViewCoverageLayer class

`google.maps.StreetViewCoverageLayer` class

A layer that illustrates the locations where Street View is available.

This class extends `MVCObject`.

Constructor	
`StreetViewCoverageLayer()`	Creates a new instance of `StreetViewCoverageLayer`.

Methods	
`getMap()`	**Return Value:** `Map` Returns the map on which this layer is displayed.
`setMap(map:Map)`	**Return Value:** None Renders the layer on the specified map. If the map is set to null, the layer will be removed.

MapsEventListener object specification

`google.maps.MapsEventListener` object specification

This class is opaque. It has no methods and no constructor. Its instances are returned from addListener(), addDomListener() and are eventually passed back to removeListener().

event namespace

`google.maps.event` namespace

Methods	
`addDomListener(instance:Object, eventName:string, handler:function(?), capture?:boolean)`	**Return Value:** `MapsEventListener` Cross browser event handler registration. This listener is removed by calling removeListener(handle) for the handle that is returned by this function.
`addDomListenerOnce( instance:Object, eventName:string, handler:function(?), capture?:boolean)`	**Return Value:** `MapsEventListener` Wrapper around addDomListener that removes the listener after the first event.
`addListener(instance:Object, eventName:string, handler:function(?))`	**Return Value:** `MapsEventListener` Adds the given listener function to the given event name for the given object instance. Returns an identifier for this listener that can be used with removeListener().
`addListenerOnce(instance:Object, eventName:string, handler:function(?))`	**Return Value:** `MapsEventListener` Like addListener, but the handler removes itself after handling the first event.
`clearInstanceListeners( instance:Object)`	**Return Value:** None Removes all listeners for all events for the given instance.
`clearListeners(instance:Object, eventName:string)`	**Return Value:** None Removes all listeners for the given event for the given instance.
`removeListener( listener:`*`MapsEventListener`*`)`	**Return Value:** None Removes the given listener, which should have been returned by addListener above.
`trigger(instance:Object, eventName:string, var_args:*)`	**Return Value:** None Triggers the given event. All arguments after eventName are passed as arguments to the listeners.

MouseEvent object specification

`google.maps.MouseEvent` object specification

This object is returned from various mouse events on the map and overlays, and contains all the fields shown below.

Methods	
`stop()`	**Return Value:** None Prevents this event from propagating further.

Properties	
`latLng`	**Type:** `LatLng` The latitude/longitude that was below the cursor when the event occurred.

LatLng class

`google.maps.LatLng` class

A `LatLng` is a point in geographical coordinates: latitude and longitude.

- Latitude ranges between -90 and 90 degrees, inclusive. Values above or below this range will be clamped to the range [-90, 90]. This means that if the value specified is less than -90, it will be set to -90. And if the value is greater than 90, it will be set to 90.
- Longitude ranges between -180 and 180 degrees, inclusive. Values above or below this range will be wrapped so that they fall within the range. For example, a value of -190 will be converted to 170. A value of 190 will be converted to -170. This reflects the fact that longitudes wrap around the globe.

Although the default map projection associates longitude with the x-coordinate of the map, and latitude with the y-coordinate, the latitude coordinate is always written *first*, followed by the longitude.
Notice that you cannot modify the coordinates of a `LatLng`. If you want to compute another point, you have to create a new one.

Constructor	
`LatLng( lat:number, lng:number, noWrap?:boolean)`	Creates a `LatLng` object representing a geographic point. Latitude is specified in degrees within the range [-90, 90]. Longitude is specified in degrees within the range [-180, 180]. Set `noWrap` to `true` to enable values outside of this range. Note the ordering of latitude and longitude.

Methods	
`equals(other:LatLng)`	**Return Value:** `boolean` Comparison function.
`lat()`	**Return Value:** `number` Returns the latitude in degrees.
`lng()`	**Return Value:** `number` Returns the longitude in degrees.
`toString()`	**Return Value:** `string` Converts to string representation.
`toUrlValue(precision?:number)`	**Return Value:** `string` Returns a string of the form "lat,lng" for this LatLng. We round the lat/lng values to 6 decimal places by default.

LatLngLiteral object specification

`google.maps.LatLngLiteral` object specification

Object literals are accepted in place of `LatLng` objects, as a convenience, in many places. These are converted to `LatLng` objects when the Maps API encounters them.

Examples:

```
map.setCenter({lat: -34, lng: 151});
new google.maps.Marker({position: {lat: -34, lng: 151}, map: map});
```

LatLng object literals are not supported in the Geometry library.

Properties	
`lat`	**Type:** `number` Latitude in degrees. Values will be clamped to the range [-90, 90]. This means that if the value specified is less than -90, it will be set to -90. And if the value is greater than 90, it will be set to 90.

specified is less than -90, it will be set to -90. And if the value is greater than 90, it will be set to 90.

`lng` **Type:** `number`

Longitude in degrees. Values outside the range [-180, 180] will be wrapped so that they fall within the range. For example, a value of -190 will be converted to 170. A value of 190 will be converted to -170. This reflects the fact that longitudes wrap around the globe.

LatLngBounds class

`google.maps.LatLngBounds` class

A `LatLngBounds` instance represents a rectangle in geographical coordinates, including one that crosses the 180 degrees longitudinal meridian.

Constructor	
`LatLngBounds(sw?:LatLng|LatLngLiteral, ne?:LatLng|LatLngLiteral)`	Constructs a rectangle from the points at its south-west and north-east corners.

Methods	
`contains( latLng:LatLng)`	**Return Value:** `boolean` Returns `true` if the given lat/lng is in this bounds.
`equals( other:LatLngBounds)`	**Return Value:** `boolean` Returns `true` if this bounds approximately equals the given bounds.
`extend( point:LatLng)`	**Return Value:** `LatLngBounds` Extends this bounds to contain the given point.
`getCenter()`	**Return Value:** `LatLng` Computes the center of this LatLngBounds
`getNorthEast()`	**Return Value:** `LatLng` Returns the north-east corner of this bounds.
`getSouthWest()`	**Return Value:** `LatLng` Returns the south-west corner of this bounds.
`intersects(`	**Return Value:** `boolean`

`other:LatLngBounds)`	Returns `true` if this bounds shares any points with this bounds.
`isEmpty()`	**Return Value:** `boolean` Returns if the bounds are empty.
`toSpan()`	**Return Value:** `LatLng` Converts the given map bounds to a lat/lng span.
`toString()`	**Return Value:** `string` Converts to string.
`toUrlValue(precision?:number)`	**Return Value:** `string` Returns a string of the form "lat_lo,lng_lo,lat_hi,lng_hi" for this bounds, where "lo" corresponds to the southwest corner of the bounding box, while "hi" corresponds to the northeast corner of that box.
`union(other:LatLngBounds)`	**Return Value:** `LatLngBounds` Extends this bounds to contain the union of this and the given bounds.

Point class

`google.maps.Point` class

Constructor	
`Point(x:number, y:number)`	A point on a two-dimensional plane.

Methods	
`equals(other:Point)`	**Return Value:** `boolean` Compares two Points
`toString()`	**Return Value:** `string` Returns a string representation of this Point.

Properties

x	**Type:** `number` The X coordinate
y	**Type:** `number` The Y coordinate

Size class

`google.maps.Size` class

Constructor	
`Size(width:number, height:number, widthUnit?:string, heightUnit?:string)`	Two-dimensonal size, where width is the distance on the x-axis, and height is the distance on the y-axis.

Methods	
`equals(other:Size)`	**Return Value:** `boolean` Compares two Sizes.
`toString()`	**Return Value:** `string` Returns a string representation of this Size.

Properties	
`height`	**Type:** `number` The height along the y-axis, in pixels.
`width`	**Type:** `number` The width along the x-axis, in pixels.

MVCObject class

`google.maps.MVCObject` class

Base class implementing KVO.

The `MVCObject` constructor is guaranteed to be an empty function, and so you may inherit from `MVCObject` by simply writing `MySubclass.prototype = new google.maps.MVCObject();`. Unless otherwise noted, this is not true of other classes in the API, and inheriting from other classes in the API is not supported.

Constructor	
`MVCObject()`	Creates an `MVCObject`.

Methods	
`addListener(eventName:string, handler:Function)`	**Return Value:** `MapsEventListener` Adds the given listener function to the given event name. Returns an identifier for this listener that can be used with `google.maps.event.removeListener`.
`bindTo(key:string, target:MVCObject, targetKey?:string, noNotify?:boolean)`	**Return Value:** None Binds a View to a Model.
`changed(key:string)`	**Return Value:** None Generic handler for state changes. Override this in derived classes to handle arbitrary state changes.
`get(key:string)`	**Return Value:** `*` Gets a value.
`notify(key:string)`	**Return Value:** None Notify all observers of a change on this property. This notifies both objects that are bound to the object's property as well as the object that it is bound to.
`set(key:string, value:*)`	**Return Value:** None Sets a value.
`setValues(values:Object|undefined)`	**Return Value:** None Sets a collection of key-value pairs.

	Sets a collection of key-value pairs.
`unbind(key:string)`	**Return Value:** None Removes a binding. Unbinding will set the unbound property to the current value. The object will not be notified, as the value has not changed.
`unbindAll()`	**Return Value:** None Removes all bindings.

MVCArray class

`google.maps.MVCArray` class

This class extends `MVCObject`.

Constructor	
`MVCArray(array?:Array)`	A mutable MVC Array.

Methods	
`clear()`	**Return Value:** None Removes all elements from the array.
`forEach(callback:function(*, number))`	**Return Value:** None Iterate over each element, calling the provided callback. The callback is called for each element like: callback(element, index).
`getArray()`	**Return Value:** `Array` Returns a reference to the underlying Array. Warning: if the Array is mutated, no events will be fired by this object.
`getAt(i:number)`	**Return Value:** `*` Returns the element at the specified index.
`getLength()`	**Return Value:** `number` Returns the number of elements in this array.

`insertAt(i:number, elem:*)`	**Return Value:** None Inserts an element at the specified index.
`pop()`	**Return Value:** `*` Removes the last element of the array and returns that element.
`push(elem:*)`	**Return Value:** `number` Adds one element to the end of the array and returns the new length of the array.
`removeAt(i:number)`	**Return Value:** `*` Removes an element from the specified index.
`setAt(i:number, elem:*)`	**Return Value:** None Sets an element at the specified index.

Events	
`insert_at`	**Arguments:** `number` This event is fired when `insertAt()` is called. The event passes the index that was passed to `insertAt()`.
`remove_at`	**Arguments:** `number, *` This event is fired when `removeAt()` is called. The event passes the index that was passed to `removeAt()` and the element that was removed from the array.
`set_at`	**Arguments:** `number, *` This event is fired when `setAt()` is called. The event passes the index that was passed to `setAt()` and the element that was previously in the array at that index.

encoding namespace

`google.maps.geometry.encoding` namespace

Utilities for polyline encoding and decoding.

Library

geometry

Methods	
`decodePath(encodedPath:string)`	**Return Value:** `Array<LatLng>` Decodes an encoded path string into a sequence of LatLngs.
`encodePath(path:Array<LatLng>|MVCArray<LatLng>)`	**Return Value:** `string` Encodes a sequence of LatLngs into an encoded path string.

spherical namespace

`google.maps.geometry.spherical` namespace

Utility functions for computing geodesic angles, distances and areas. The default radius is Earth's radius of 6378137 meters.

Library

geometry

Methods	
`computeArea(path:Array<LatLng>|MVCArray<LatLng>, radius?:number)`	**Return Value:** `number` Returns the area of a closed path. The computed area uses the same units as the radius. The radius defaults to the Earth's radius in meters, in which case the area is in square meters.
`computeDistanceBetween(from:LatLng, to:LatLng, radius?:number)`	**Return Value:** `number` Returns the distance, in meters, between two LatLngs. You can optionally specify a custom radius. The radius defaults to the radius of the Earth.
`computeHeading(from:LatLng, to:LatLng)`	**Return Value:** `number` Returns the heading from one LatLng to another LatLng. Headings are expressed in degrees clockwise from North within the range [-180,180).
`computeLength(path:Array<LatLng>|`	**Return Value:** `number` Returns the length of the given path.

MVCArray<LatLng>, radius?:number)	
computeOffset(from:LatLng, distance:number, heading:number, radius?:number)	**Return Value:** LatLng Returns the LatLng resulting from moving a distance from an origin in the specified heading (expressed in degrees clockwise from north).
computeOffsetOrigin(to:LatLng, distance:number, heading:number, radius?:number)	**Return Value:** LatLng Returns the location of origin when provided with a LatLng destination, meters travelled and original heading. Headings are expressed in degrees clockwise from North. This function returns null when no solution is available.
computeSignedArea(loop:Array<LatLng>|MVCArray<LatLng>, radius?:number)	**Return Value:** number Returns the signed area of a closed path. The signed area may be used to determine the orientation of the path. The computed area uses the same units as the radius. The radius defaults to the Earth's radius in meters, in which case the area is in square meters.
interpolate(from:LatLng, to:LatLng, fraction:number)	**Return Value:** LatLng Returns the LatLng which lies the given fraction of the way between the origin LatLng and the destination LatLng.

poly namespace

`google.maps.geometry.poly` namespace

Utility functions for computations involving polygons and polylines.

Library

geometry

Methods	
containsLocation(point:LatLng, polygon:Polygon)	**Return Value:** boolean Computes whether the given point lies inside the specified polygon.
isLocationOnEdge(point:LatLng,	**Return Value:** boolean Computes whether the given point lies on or near to a polyline, or the edge of a

`poly:Polygon|Polyline, tolerance?:number)`	Computes whether the given point lies on or near to a polyline, or the edge of a polygon, within a specified tolerance. Returns `true` when the difference between the latitude and longitude of the supplied point, and the closest point on the edge, is less than the tolerance. The tolerance defaults to 10^{-9} degrees.

AdUnit class

`google.maps.adsense.AdUnit` class

Implements AdSense for Content advertising on an associated map. To use an `AdUnit`, you must obtain and specify an AdSense for Content publisher ID within the `AdUnit`'s constructor options.

This class extends `MVCObject`.

Library

adsense

Constructor	
`AdUnit(container:Node, opts:AdUnitOptions)`	Creates an AdSense for Content display ad on the associated map.

Methods	
`getBackgroundColor()`	**Return Value:** `string` Returns the AdUnit's background color.
`getBorderColor()`	**Return Value:** `string` Returns the AdUnit's border color.
`getChannelNumber()`	**Return Value:** `string` Returns the channel number in use by this `AdUnit`.
`getContainer()`	**Return Value:** Node Returns the containing element of the AdUnit.
`getFormat()`	**Return Value:** `AdFormat`

	Returns the format in use by this AdUnit.
getMap()	**Return Value:** Map Returns the map to which this AdUnit's ads are targeted.
getPosition()	**Return Value:** ControlPosition Returns the ControlPosition at which this AdUnit is displayed on the map.
getPublisherId()	**Return Value:** string Returns the specified AdSense For Content publisher ID.
getTextColor()	**Return Value:** string Returns the AdUnit's text color.
getTitleColor()	**Return Value:** string Returns the AdUnit's title color.
getUrlColor()	**Return Value:** string Returns the AdUnit's URL color.
setBackgroundColor(backgroundColor:string)	**Return Value:** None Sets the AdUnit's background color.
setBorderColor(borderColor:string)	**Return Value:** None Sets the AdUnit's border color.
setChannelNumber(channelNumber:string)	**Return Value:** None Specifies the channel number for this AdUnit. Channel numbers are optional and can be created for Google AdSense tracking.
setFormat(format:AdFormat)	**Return Value:** None Specifies the display format for this AdUnit.
setMap(map:Map)	**Return Value:** None Associates this AdUnit with the specified map. Ads will be targeted to the map's viewport. The map must be specified in order to display ads.
setPosition(position:ControlPosition)	**Return Value:** None Sets the ControlPosition at which to display the AdUnit on the map. If the position is set to null, the AdUnit is removed from the map.
setTextColor(	**Return Value:** None

`setTextColor(`
`textColor:string)`
	Return Value: None
	Sets the AdUnit's text color.

`setTitleColor(`
`titleColor:string)`
	Return Value: None
	Sets the AdUnit's title color.

`setUrlColor(`
`urlColor:string)`
	Return Value: None
	Sets the AdUnit's URL color.

AdUnitOptions object specification

`google.maps.adsense.AdUnitOptions` object specification

Library

adsense

Properties	
`backgroundColor`	**Type:** `string` The AdUnit's background color. (Optional)
`borderColor`	**Type:** `string` The AdUnit's border color. (Optional)
`channelNumber`	**Type:** `string` The AdSense For Content channel number for tracking the performance of this `AdUnit`. It must be stored as a string as it will typically be a large UINT64. (Optional)
`format`	**Type:** `AdFormat` the Format of the `AdUnit`. See https://google.com/adsense/adformats. (Optional)
`map`	**Type:** `Map` The map associated with this `AdUnit`. Ads will be targeted to the location the map's viewport. (Required)
`position`	**Type:** `ControlPosition` The position of the `AdUnit`. If specified, the `AdUnit` will be displayed at this position. Otherwise, it will not be added to the map. (Optional)

`publisherId`	**Type:** `string` Your AdSense for Content publisher ID. Required and must be set at the time of initialization. (Required)
`textColor`	**Type:** `string` The `AdUnit`'s text color. (Optional)
`titleColor`	**Type:** `string` The `AdUnit`'s title color. (Optional)
`urlColor`	**Type:** `string` The `AdUnit`'s URL color. (Optional)

AdFormat class

`google.maps.adsense.AdFormat` class

Identifiers used to specify an AdSense For Content format. See https://google.com/adsense/adformats.

Library

adsense

Constant	
`BANNER`	A horizontal "banner" ad. (468x60px)
`BUTTON`	A small ad. (125x125px)
`HALF_BANNER`	A smaller horizontal "banner" ad. (234x60px)
`LARGE_HORIZONTAL_LINK_UNIT`	A large horizontal ad link unit. (728x15px)
`LARGE_RECTANGLE`	A large rectangular ad. (336x280px)
`LARGE_VERTICAL_LINK_UNIT`	A large vertical ad link unit. (180x90px)
`LEADERBOARD`	A fully horizontal display area. (728x90px)
`MEDIUM_RECTANGLE`	A medium rectangular ad. (300x250px)

`MEDIUM_VERTICAL_LINK_UNIT`	A medium vertical ad link unit. (160x90px)
`SKYSCRAPER`	A large vertical ad. (120x600px)
`SMALL_HORIZONTAL_LINK_UNIT`	A small horizontal ad link unit. (468x15px)
`SMALL_RECTANGLE`	A small rectangular ad. (180x150px)
`SMALL_SQUARE`	A smaller square ad. (200x200px)
`SMALL_VERTICAL_LINK_UNIT`	A small vertical ad link unit. (120x90px)
`SQUARE`	A square ad with large type. (250x250px)
`VERTICAL_BANNER`	A medium-sized vertical ad. (120x240px)
`WIDE_SKYSCRAPER`	A wide, vertical ad using larger type. (160x600px)
`X_LARGE_VERTICAL_LINK_UNIT`	An extra large vertical ad link unit. (200x90px)

Autocomplete class

`google.maps.places.Autocomplete` class

A service to provide Place predictions based on a user's text input. It attaches to an input element of type `text`, and listens for text entry in that field. The list of predictions is presented as a drop-down list, and is updated as text is entered.

This class extends `MVCObject`.

Library

places

Constructor	
`Autocomplete( inputField:HTMLInputElement, opts?:`*`AutocompleteOptions`*`)`	Creates a new instance of `Autocomplete` that attaches to the specified input text field with the given options.

Methods	
`getBounds()`	**Return Value:** LatLngBounds

`getBounds()`	**Return Value:** LatLngBounds Returns the bounds to which predictions are biased.
`getPlace()`	**Return Value:** `PlaceResult` Returns the details of the Place selected by user if the details were successfully retrieved. Otherwise returns a stub Place object, with the name property set to the current value of the input field.
`setBounds(bounds:LatLngBounds)`	**Return Value:** None Sets the preferred area within which to return Place results. Results are biased towards, but not restricted to, this area.
`setComponentRestrictions(restrictions:`*`ComponentRestrictions`*`)`	**Return Value:** None Sets the component restrictions. Component restrictions are used to restrict predictions to only those within the parent component. E.g., the country.
`setTypes(types:Array<string>)`	**Return Value:** None Sets the types of predictions to be returned. For a list of supported types, see the developer's guide. If no type is specified, all types will be returned. The `setTypes` method accepts a single element array.

Events	
`place_changed`	**Arguments:** None This event is fired when a `PlaceResult` is made available for a Place the user has selected. If the user enters the name of a Place that was not suggested by the control and presses the Enter key, or if a Place detail request fails, a `place_changed` event will be fired that contains the user input in the name property, with no other properties defined.

AutocompleteOptions object specification

`google.maps.places.AutocompleteOptions` object specification

The options that can be set on an `Autocomplete` object.

Library

places

Properties	
`bounds`	**Type:** `LatLngBounds` The area in which to search for places. Results are biased towards, but not restricted to, places contained within these bounds.
`componentRestrictions`	**Type:** `ComponentRestrictions` The component restrictions. Component restrictions are used to restrict predictions to only those within the parent component. E.g., the country.
`types`	**Type:** `Array<string>` The types of predictions to be returned. For a list of supported types, see the developer's guide. If nothing is specified, all types are returned. In general only a single type is allowed. The exception is that you can safely mix the `'geocode'` and `'establishment'` types, but note that this will have the same effect as specifying no types.

AutocompletePrediction object specification

`google.maps.places.AutocompletePrediction` object specification

Library

places

Properties	
`description`	**Type:** `string` This is the unformatted version of the query suggested by the Places service.
`matched_substrings`	**Type:** `Array<PredictionSubstring>` A set of substrings in the place's description that match elements in the user's input, suitable for use in highlighting those substrings. Each substring is identified

	by an offset and a length, expressed in unicode characters.
`place_id`	**Type:** `string` A place ID that can be used to retrieve details about this place using the place details service (see `PlacesService.getDetails()`).
`terms`	**Type:** `Array<PredictionTerm>` Information about individual terms in the above description, from most to least specific. For example, "Taco Bell", "Willitis", and "CA".
`types`	**Type:** `Array<string>` An array of types that the prediction belongs to, for example `'establishment'` or `'geocode'`.

PredictionTerm object specification

`google.maps.places.PredictionTerm` object specification

Library

places

Properties	
`offset`	**Type:** `number` The offset, in unicode characters, of the start of this term in the description of the place.
`value`	**Type:** `string` The value of this term, e.g. "Taco Bell".

PredictionSubstring object specification

`google.maps.places.PredictionSubstring` object specification

Library

places

Properties	
`length`	**Type:** `number` The length of the substring.
`offset`	**Type:** `number` The offset to the substring's start within the description string.

AutocompleteService class

`google.maps.places.AutocompleteService` class

Contains methods related to retrieving Autocomplete predictions.

Library

places

Constructor	
`AutocompleteService()`	Creates a new instance of the `AutocompleteService`.

Methods	
`getPlacePredictions(request:`*`AutocompletionRequest`*`, callback:function(Array<AutocompletePrediction>, PlacesServiceStatus))`	**Return Value:** None Retrieves place autocomplete predictions based on the supplied autocomplete request.
`getQueryPredictions( request:`*`QueryAutocompletionRequest`*`, callback:function( Array<QueryAutocompletePrediction>, PlacesServiceStatus))`	**Return Value:** None Retrieves query autocomplete predictions based on the supplied query autocomplete request.

AutocompletionRequest object specification

`google.maps.places.AutocompletionRequest` object specification

An Autocompletion request to be sent to the `AutocompleteService`.

Library

places

Properties	
`bounds`	**Type:** `LatLngBounds` Bounds for prediction biasing. Predictions will be biased towards, but not restricted to, the given bounds. Both `location` and `radius` will be ignored if bounds is set.
`componentRestrictions`	**Type:** `ComponentRestrictions` The component restrictions. Component restrictions are used to restrict predictions to only those within the parent component. E.g., the country.
`input`	**Type:** `string` The user entered input string.
`location`	**Type:** `LatLng` Location for prediction biasing. Predictions will be biased towards the given `location` and `radius`. Alternatively, bounds can be used.
`offset`	**Type:** `number` The character position in the input term at which the service uses text for predictions (the position of the cursor in the input field).
`radius`	**Type:** `number` The radius of the area used for prediction biasing. The `radius` is specified in meters, and must always be accompanied by a `location` property. Alternatively, bounds can be used.
`types`	**Type:** `Array<string>` The types of predictions to be returned. Four types are supported: `'establishment'` for businesses, `'geocode'` for addresses, `'(regions)'` for administrative regions and `'(cities)'` for localities. If nothing is specified, all types are returned.

ComponentRestrictions object specification

`google.maps.places.ComponentRestrictions` object specification

Defines the component restrictions that can be used with the autocomplete service.

Library

places

Properties	
`country`	**Type:** `string` Restricts predictions to the specified country (ISO 3166-1 Alpha-2 country code, case insensitive). E.g., us, br, au.

PlaceAspectRating object specification

`google.maps.places.PlaceAspectRating` object specification

Defines information about an aspect of the place that users have reviewed.

Library

places

Properties	
`rating`	**Type:** `number` The rating of this aspect. For individual reviews this is an integer from 0 to 3. For aggregated ratings of a place this is an integer from 0 to 30.
`type`	**Type:** `string` The aspect type, e.g. "food", "decor", "service", "overall".

PlaceDetailsRequest object specification

`google.maps.places.PlaceDetailsRequest` object specification

A Place details query to be sent to the `PlacesService`.

Library

places

Properties	
`placeId`	**Type:** `string` The Place ID of the Place for which details are being requested.

PlaceGeometry object specification

`google.maps.places.PlaceGeometry` object specification

Defines information about the geometry of a Place.

Library

places

Properties	
`location`	**Type:** `LatLng` The Place's position.
`viewport`	**Type:** `LatLngBounds` The preferred viewport when displaying this Place on a map. This property will be null if the preferred viewport for the Place is not known.

PlacePhoto object specification

`google.maps.places.PlacePhoto` object specification

Represents a photo element of a Place.

Library

places

Methods	

`getUrl(`
`opts:`*`PhotoOptions`*`)`
	Return Value: `string`
	Returns the image URL corresponding to the specified options. You must include a PhotoOptions object with at least one of maxWidth or maxHeight specified.

Properties	
`height`	**Type:** `number` The height of the photo in pixels.
`html_attributions`	**Type:** `Array<string>` Attribution text to be displayed for this photo.
`width`	**Type:** `number` The width of the photo in pixels.

PhotoOptions object specification

`google.maps.places.PhotoOptions` object specification

Defines photo-requesting options.

Library

places

Properties	
`maxHeight`	**Type:** `number`

`maxHeight`	**Type:** `number` The maximum height in pixels of the returned image.
`maxWidth`	**Type:** `number` The maximum width in pixels of the returned image.

PlaceResult object specification

`google.maps.places.PlaceResult` object specification

Defines information about a Place.

Library

places

Properties	
`address_components`	**Type:** `Array<GeocoderAddressComponent>` The collection of address components for this Place's location.
`aspects`	**Type:** `Array<PlaceAspectRating>` The rated aspects of this Place, based on Google and Zagat user reviews. The ratings are on a scale of 0 to 30.
`formatted_address`	**Type:** `string` The Place's full address.
`formatted_phone_number`	**Type:** `string` The Place's phone number, formatted according to the number's regional convention.
`geometry`	**Type:** `PlaceGeometry` The Place's geometry-related information.
`html_attributions`	**Type:** `Array<string>` Attribution text to be displayed for this Place result.
`icon`	**Type:** `string` URL to an image resource that can be used to represent this Place's

	URL to an image resource that can be used to represent this Place's category.
`international_phone_number`	**Type:** `string` The Place's phone number in international format. International format includes the country code, and is prefixed with the plus (+) sign.
`name`	**Type:** `string` The Place's name. Note: In the case of user entered Places, this is the raw text, as typed by the user. Please exercise caution when using this data, as malicious users may try to use it as a vector for code injection attacks (See http://en.wikipedia.org/wiki/Code_injection).
`permanently_closed`	**Type:** `boolean` A flag indicating whether the Place is permanently closed. If the place is not permanently closed, the flag is not present in search or details responses.
`photos`	**Type:** `Array<PlacePhoto>` Photos of this Place. The collection will contain up to ten `PlacePhoto` objects.
`place_id`	**Type:** `string` A unique identifier for a place.
`price_level`	**Type:** `number` The price level of the Place, on a scale of 0 to 4. Price levels are interpreted as follows: `0`: Free `1`: Inexpensive `2`: Moderate `3`: Expensive `4`: Very Expensive
`rating`	**Type:** `number` A rating, between 1.0 to 5.0, based on user reviews of this Place.
`reviews`	**Type:** `Array<PlaceReview>` A list of reviews of this Place.
`types`	**Type:** `Array<string>` An array of types for this Place (e.g., `["political", "locality"]` or `["restaurant", "establishment"]`).

`url`	**Type:** `string` URL of the official Google page for this place. This will be the establishment's Google+ page if the Google+ page exists, otherwise it will be the Google-owned page that contains the best available information about the place.
`utc_offset`	**Type:** `number` The offset from UTC of the Place's current timezone, in minutes. For example, Sydney, Australia in daylight savings is 11 hours ahead of UTC, so the utc_offset will be 660. For timezones behind UTC, the offset is negative. For example, utc_offest is -60 for Cape Verde.
`vicinity`	**Type:** `string` A fragment of the Place's address for disambiguation (usually street name and locality).
`website`	**Type:** `string` The authoritative website for this Place, such as a business' homepage.

PlaceReview object specification

`google.maps.places.PlaceReview` object specification

Represents a single review of a place.

Library

places

Properties	
`aspects`	**Type:** `Array<PlaceAspectRating>` The aspects rated by the review. The ratings on a scale of 0 to 3.
`author_name`	**Type:** `string` The name of the reviewer.
`author_url`	**Type:** `string`

	A link to the reviewer's profile. This will be undefined when the reviewer's profile is unavailable.
`language`	**Type:** `string` An IETF language code indicating the language in which this review is written. Note that this code includes only the main language tag without any secondary tag indicating country or region. For example, all the English reviews are tagged as 'en' rather than 'en-AU' or 'en-UK'.
`text`	**Type:** `string` The text of a review.

PlaceSearchPagination object specification

`google.maps.places.PlaceSearchPagination` object specification

An object used to fetch additional pages of Places results.

Library

places

Methods	
`nextPage()`	**Return Value:** None Fetches the next page of results. Uses the same callback function that was provided to the first search request.

Properties	
`hasNextPage`	**Type:** `boolean` Indicates if further results are available. `true` when there is an additional results page.

PlaceSearchRequest object specification

`google.maps.places.PlaceSearchRequest` object specification

A Place search query to be sent to the `PlacesService`.

Library

places

Properties	
`bounds`	**Type:** `LatLngBounds` The bounds within which to search for Places. Both `location` and `radius` will be ignored if bounds is set.
`keyword`	**Type:** `string` A term to be matched against all available fields, including but not limited to name, type, and address, as well as customer reviews and other third-party content.
`location`	**Type:** `LatLng`|`LatLngLiteral` The location around which to search for Places.
`maxPriceLevel`	**Type:** `number` Restricts results to only those places at the specified price level or lower. Valid values are in the range from 0 (most affordable) to 4 (most expensive), inclusive. Must be greater than or equal to `minPrice` , if specified.
`minPriceLevel`	**Type:** `number` Restricts results to only those places at the specified price level or higher. Valid values are in the range from 0 (most affordable) to 4 (most expensive), inclusive. Must be less than or equal to `maxPrice`, if specified.
`name`	**Type:** `string` Restricts the Place search results to Places that include this text in the name.
`openNow`	**Type:** `boolean` Restricts results to only those places that are open right now.
`radius`	**Type:** `number` The distance from the given location within which to search for Places, in meters. The maximum allowed value is 50 000.
`rankBy`	**Type:** `RankBy` Specifies the ranking method to use when returning results.

	Specifies the ranking method to use when returning results.
`types`	**Type:** `Array<string>` Restricts the Place search results to Places with a type matching at least one of the specified types in this array. Valid types are given here.

PlacesService class

`google.maps.places.PlacesService` class

Contains methods related to searching for Places and retrieving details about a Place.

Library

places

Constructor	
`PlacesService( attrContainer:HTMLDivElement|Map)`	Creates a new instance of the `PlacesService` that renders attributions in the specified container.

Methods	
`getDetails( request:`*`PlaceDetailsRequest`*`, callback:function( PlaceResult, PlacesServiceStatus))`	**Return Value:** None Retrieves details about the Place identified by the given `placeId`.
`nearbySearch( request:`*`PlaceSearchRequest`*`, callback:function( Array<PlaceResult>, PlacesServiceStatus, PlaceSearchPagination))`	**Return Value:** None Retrieves a list of Places in a given area. The `PlaceResults` passed to the callback are stripped-down versions of a full PlaceResult. A more detailed `PlaceResult` for each Place can be obtained by sending a Place Details request with the desired Place's `placeId` value. The `PlaceSearchPagination` object can be used to fetch additional pages of results (null if this is the last page of results or if there is only one page of results).
`radarSearch( request:`*`RadarSearchRequest`*	**Return Value:** None

`request:RadarSearchRequest,`
`callback:function(`
`Array<PlaceResult>,`
`PlacesServiceStatus))`
	Similar to the `nearbySearch` function, with the following differences: the search response will include up to 200 Places, identified only by their geographic coordinates and `place_id`.

`textSearch(`
`request:TextSearchRequest,`
`callback:function(`
`Array<PlaceResult>,`
`PlacesServiceStatus,`
`PlaceSearchPagination))`
	Return Value: None
	Similar to the `nearbySearch` function, with the following differences: it retrieves a list of Places based on the `query` attribute in the given request object; bounds or `location + radius` parameters are optional; and the region, when provided, will not restrict the results to places inside the area, only bias the response towards results near it. The `PlaceSearchPagination` object can be used to fetch additional pages of results (null if this is the last page of results or if there is only one page of results).

PlacesServiceStatus class

`google.maps.places.PlacesServiceStatus` class

The status returned by the `PlacesService` on the completion of its searches.

Library

places

Constant	
`INVALID_REQUEST`	This request was invalid.
`OK`	The response contains a valid result.
`OVER_QUERY_LIMIT`	The application has gone over its request quota.
`REQUEST_DENIED`	The application is not allowed to use the `PlacesService`.
`UNKNOWN_ERROR`	The `PlacesService` request could not be processed due to a server error. The request may succeed if you try again.
`ZERO_RESULTS`	No result was found for this request.

QueryAutocompletePrediction object specification

`google.maps.places.QueryAutocompletePrediction` object specification

Represents a single Query Autocomplete prediction.

Library

places

Properties	
`description`	**Type:** `string` This is the unformatted version of the query suggested by the Places service.
`matched_substrings`	**Type:** `Array<PredictionSubstring>` A set of substrings in the place's description that match elements in the user's input, suitable for use in highlighting those substrings. Each substring is identified by an offset and a length, expressed in unicode characters.
`place_id`	**Type:** `string` Only available if prediction is a place. A place ID that can be used to retrieve details about this place using the place details service (see `PlacesService.getDetails()`).
`terms`	**Type:** `Array<PredictionTerm>` Information about individual terms in the above description. Categorical terms come first (e.g., "restaurant"). Address terms appear from most to least specific. For example, "San Francisco", and "CA".

QueryAutocompletionRequest object specification

`google.maps.places.QueryAutocompletionRequest` object specification

An QueryAutocompletion request to be sent to the `QueryAutocompleteService`.

Library

places

Properties	
`bounds`	**Type:** `LatLngBounds` Bounds for prediction biasing. Predictions will be biased towards, but not restricted to, the given bounds. Both `location` and `radius` will be ignored if bounds is set.
`input`	**Type:** `string` The user entered input string.
`location`	**Type:** `LatLng` Location for prediction biasing. Predictions will be biased towards the given `location` and `radius`. Alternatively, bounds can be used.
`offset`	**Type:** `number` The character position in the input term at which the service uses text for predictions (the position of the cursor in the input field).
`radius`	**Type:** `number` The radius of the area used for prediction biasing. The `radius` is specified in meters, and must always be accompanied by a `location` property. Alternatively, bounds can be used.

RadarSearchRequest object specification

`google.maps.places.RadarSearchRequest` object specification

A Radar Search request to be sent to the `PlacesService`.

Library

places

Properties	
`bounds`	**Type:** `LatLngBounds` Bounds used to bias results when searching for Places (optional). Both `location` and `radius` will be ignored if `bounds` is set. Results will not be restricted to those inside these bounds; but, results inside it will rank higher.
`keyword`	**Type:** `string` A term to be matched against all available fields, including but not limited to name, type, and

	A term to be matched against all available fields, including but not limited to name, type, and address, as well as customer reviews and other third-party content.
`location`	**Type:** `LatLng`|`LatLngLiteral` The center of the area used to bias results when searching for Places.
`name`	**Type:** `string` Restricts results to Places that include this text in the name.
`radius`	**Type:** `number` The radius of the area used to bias results when searching for Places, in meters.
`types`	**Type:** `Array<string>` Restricts the Place search results to Places with a type matching at least one of the specified types in this array. Valid types are given here.

RankBy class

`google.maps.places.RankBy` class

Ranking options for a PlaceSearchRequest.

Library

places

Constant	
`DISTANCE`	Ranks place results by distance from the location.
`PROMINENCE`	Ranks place results by their prominence.

SearchBox class

`google.maps.places.SearchBox` class

A service to provide query predictions based on a user's text input. It attaches to an input element of type `text`, and listens for text entry in that field. The list of predictions is presented as a drop-down list, and is updated as text is entered.

This class extends `MVCObject`.

Library

places

Constructor	
`SearchBox( inputField:HTMLInputElement, opts?:SearchBoxOptions)`	Creates a new instance of `SearchBox` that attaches to the specified input text field with the given options.

Methods	
`getBounds()`	**Return Value:** `LatLngBounds` Returns the bounds to which query predictions are biased.
`getPlaces()`	**Return Value:** `Array<PlaceResult>` Returns the query selected by the user, or `null` if no places have been found yet, to be used with `places_changed` event.
`setBounds( bounds:LatLngBounds)`	**Return Value:** None Sets the region to use for biasing query predictions. Results will only be biased towards this area and not be completely restricted to it.

Events	
`places_changed`	**Arguments:** None This event is fired when the user selects a query, `getPlaces` should be used to get new places.

SearchBoxOptions object specification

`google.maps.places.SearchBoxOptions` object specification

The options that can be set on a `SearchBox` object.

Library

places

Properties	
`bounds`	**Type:** `LatLngBounds` The area towards which to bias query predictions. Predictions are biased towards, but not restricted to, queries targeting these bounds.

TextSearchRequest object specification

`google.maps.places.TextSearchRequest` object specification

A text search request to be sent to the `PlacesService`.

Library

places

Properties	
`bounds`	**Type:** `LatLngBounds` Bounds used to bias results when searching for Places (optional). Both `location` and `radius` will be ignored if `bounds` is set. Results will not be restricted to those inside these bounds; but, results inside it will rank higher.
`location`	**Type:** `LatLng`|`LatLngLiteral` The center of the area used to bias results when searching for Places.
`query`	**Type:** `string` The request's query term. e.g. the name of a place ('Eiffel Tower'), a category followed by the name of a location ('pizza in New York'), or the name of a place followed by a location disambiguator ('Starbucks in Sydney').
`radius`	**Type:** `number` The radius of the area used to bias results when searching for Places, in meters.

`types`	**Type:** `Array<string>` Restricts the Place search results to Places with a type matching at least one of the specified types in this array. Valid types are given here.

DrawingManager class

`google.maps.drawing.DrawingManager` class

Allows users to draw markers, polygons, polylines, rectangles, and circles on the map. The `DrawingManager`'s drawing mode defines the type of overlay that will be created by the user. Adds a control to the map, allowing the user to switch drawing mode.

This class extends `MVCObject`.

Library

drawing

Constructor	
`DrawingManager(options?:`*`DrawingManagerOptions`*`)`	Creates a `DrawingManager` that allows users to draw overlays on the map, and switch between the type of overlay to be drawn with a drawing control.

Methods	
`getDrawingMode()`	**Return Value:** `OverlayType` Returns the `DrawingManager`'s drawing mode.
`getMap()`	**Return Value:** `Map` Returns the `Map` to which the `DrawingManager` is attached, which is the `Map` on which the overlays created will be placed.
`setDrawingMode(drawingMode:OverlayType)`	**Return Value:** None Changes the `DrawingManager`'s drawing mode, which defines the type of overlay to be added on the map. Accepted values are `MARKER`, `POLYGON`, `POLYLINE`, `RECTANGLE`, `CIRCLE`, or `null`. A drawing mode of `null` means that the user can interact with the

	map as normal, and clicks do not draw anything.
`setMap(map:Map)`	**Return Value:** None Attaches the `DrawingManager` object to the specified `Map`.

`setOptions(`
`options:`*`DrawingManagerOptions`*`)`
	Return Value: None
	Sets the `DrawingManager`'s options.

Events	
`circlecomplete`	**Arguments:** `Circle` This event is fired when the user has finished drawing a circle.
`markercomplete`	**Arguments:** `Marker` This event is fired when the user has finished drawing a marker.
`overlaycomplete`	**Arguments:** *`OverlayCompleteEvent`* This event is fired when the user has finished drawing an overlay of any type.
`polygoncomplete`	**Arguments:** `Polygon` This event is fired when the user has finished drawing a polygon.
`polylinecomplete`	**Arguments:** `Polyline` This event is fired when the user has finished drawing a polyline.
`rectanglecomplete`	**Arguments:** `Rectangle` This event is fired when the user has finished drawing a rectangle.

DrawingManagerOptions object specification

`google.maps.drawing.DrawingManagerOptions` object specification

Options for the drawing manager.

Library

drawing

Properties	
`circleOptions`	**Type:** `CircleOptions` Options to apply to any new circles created with this `DrawingManager`. The `center` and `radius` properties are ignored, and the `map` property of a new circle is always set to the `DrawingManager`'s map.
`drawingControl`	**Type:** `boolean` The enabled/disabled state of the drawing control. Defaults to `true`.
`drawingControlOptions`	**Type:** `DrawingControlOptions` The display options for the drawing control.
`drawingMode`	**Type:** `OverlayType` The `DrawingManager`'s drawing mode, which defines the type of overlay to be added on the map. Accepted values are `MARKER`, `POLYGON`, `POLYLINE`, `RECTANGLE`, `CIRCLE`, or `null`. A drawing mode of `null` means that the user can interact with the map as normal, and clicks do not draw anything.
`map`	**Type:** `Map` The `Map` to which the `DrawingManager` is attached, which is the `Map` on which the overlays created will be placed.
`markerOptions`	**Type:** `MarkerOptions` Options to apply to any new markers created with this `DrawingManager`. The `position` property is ignored, and the `map` property of a new marker is always set to the `DrawingManager`'s map.
`polygonOptions`	**Type:** `PolygonOptions` Options to apply to any new polygons created with this `DrawingManager`. The `paths` property is ignored, and the `map` property of a new polygon is always set to the `DrawingManager`'s map.
`polylineOptions`	**Type:** `PolylineOptions` Options to apply to any new polylines created with this `DrawingManager`. The `path` property is ignored, and the `map` property of a new polyline is always set to the `DrawingManager`'s map.
`rectangleOptions`	**Type:** `RectangleOptions` Options to apply to any new rectangles created with this `DrawingManager`. The `bounds` property is ignored, and the `map` property of a new rectangle is always set to the `DrawingManager`'s map.

DrawingControlOptions object specification

`google.maps.drawing.DrawingControlOptions` object specification

Options for the rendering of the drawing control.

Library

drawing

Properties	
`drawingModes`	**Type:** `Array<OverlayType>` The drawing modes to display in the drawing control, in the order in which they are to be displayed. The hand icon (which corresponds to the null drawing mode) is always available and is not to be specified in this array. Defaults to `[MARKER, POLYLINE, RECTANGLE, CIRCLE, POLYGON]`.
`position`	**Type:** `ControlPosition` Position id. Used to specify the position of the control on the map. The default position is `TOP_LEFT`.

OverlayCompleteEvent object specification

`google.maps.drawing.OverlayCompleteEvent` object specification

The properties of an overlaycomplete event on a `DrawingManager`.

Library

drawing

Properties	
`overlay`	**Type:** `Marker|Polygon|Polyline|Rectangle|Circle` The completed overlay.
`type`	**Type:** `OverlayType` The completed overlay's type.

OverlayType class

`google.maps.drawing.OverlayType` class

The types of overlay that may be created by the `DrawingManager`.

Library

drawing

Constant	
`CIRCLE`	Specifies that the `DrawingManager` creates circles, and that the overlay given in the `overlaycomplete` event is a circle.
`MARKER`	Specifies that the `DrawingManager` creates markers, and that the overlay given in the `overlaycomplete` event is a marker.
`POLYGON`	Specifies that the `DrawingManager` creates polygons, and that the overlay given in the `overlaycomplete` event is a polygon.
`POLYLINE`	Specifies that the `DrawingManager` creates polylines, and that the overlay given in the `overlaycomplete` event is a polyline.
`RECTANGLE`	Specifies that the `DrawingManager` creates rectangles, and that the overlay given in the `overlaycomplete` event is a rectangle.

MapsEngineLayer class

`google.maps.visualization.MapsEngineLayer` class

A `MapsEngineLayer` allows you to display data from Google Maps Engine or the Google Earth Gallery.

This class extends `MVCObject`.

Library

Constructor	
`MapsEngineLayer(options:`*`MapsEngineLayerOptions`*`)`	Creates a new instance of `MapsEngineLayer`.

Methods	
`getLayerId()`	**Return Value:** `string` Returns the ID of the Maps Engine layer being displayed, if set.
`getLayerKey()`	**Return Value:** `string` Returns the key of the layer to be displayed.
`getMap()`	**Return Value:** `Map` Returns the map on which this layer is displayed.
`getMapId()`	**Return Value:** `string` Returns the ID of the Maps Engine map to which the layer belongs.
`getOpacity()`	**Return Value:** `number` Returns the opacity of the layer. Applies only to imagery layers.
`getProperties()`	**Return Value:** `MapsEngineLayerProperties` Returns properties of the Maps Engine layer, which are available once the layer has loaded.
`getStatus()`	**Return Value:** `MapsEngineStatus` Returns the status of the layer, which is available once the requested layer has loaded.
`getZIndex()`	**Return Value:** `number` Returns the z-index.
`setLayerId(layerId:string)`	**Return Value:** None Sets the ID of a single Maps Engine layer to display. Changing this value will cause the layer to be redrawn.
`setLayerKey(layerKey:string)`	**Return Value:** None

	Sets the key of the layer to be displayed. Maps Engine layer keys are only unique within a single map, and can be changed by map owners. Changing this value will cause the layer to be redrawn.
`setMap(map:Map)`	**Return Value:** None Renders the layer on the specified map. If map is set to null, the layer will be removed.
`setMapId(mapId:string)`	**Return Value:** None Sets the ID of the Maps Engine map that contains the layer with the given `layerKey`. Changing this value will cause the layer to be redrawn.
`setOpacity(opacity:number)`	**Return Value:** None Sets the opacity of the layer, expressed as a number between 0 and 1. Applies only to imagery layers. Note: Be careful of setting this option for other layer types, as it may become effective in the future.
`setOptions(options:`*`MapsEngineLayerOptions`*`)`	**Return Value:** None
`setZIndex(zIndex:number)`	**Return Value:** None Sets the z-index. Only applies to Vector and KML layers.

Events	
`click`	**Arguments:** *`MapsEngineMouseEvent`* This event is fired when a feature in the layer is clicked.
`properties_changed`	**Arguments:** None This event is fired when the layer has finished loading, and the layer's properties are available for reading.
`status_changed`	**Arguments:** None This event is fired when the layer has finished loading, and the status is available to determine if the layer loaded successfully.

MapsEngineLayerOptions object specification

`google.maps.visualization.MapsEngineLayerOptions` object specification

This object defines the properties that can be set on a `MapsEngineLayer` object. `layerId`, or both `mapId` and `layerKey`, must be set.

Library

visualization

Properties	
`accessToken`	**Type:** `string` The authentication token returned by an OAuth 2.0 authentication request.
`clickable`	**Type:** `boolean` If true, the layer receives mouse events. Default value is true.
`fitBounds`	**Type:** `boolean` If this option is set to true, the map viewport is centered and zoomed to the bounding box of the contents of the layer. Default value is false. Applies only to KML layers. Note: Be careful when setting this option for other layer types, as it may become effective in the future.
`layerId`	**Type:** `string` The ID of a single Maps Engine layer to display.
`layerKey`	**Type:** `string` The key of the layer to display. Maps Engine layer keys are only unique within a single map, and can be changed by map owners.
`map`	**Type:** `Map` The map on which to display the layer.
`mapId`	**Type:** `string` The ID of the Maps Engine map that contains the layer with the given `layerKey`.
`opacity`	**Type:** `number` The opacity of the layer, expressed as a number between 0 and 1. Defaults to 1. Applies only to imagery layers. Note: Be careful of setting this option for other

	layer types, as it may become effective in the future.
`suppressInfoWindows`	**Type:** `boolean` Suppress the rendering of info windows when layer features are clicked.
`zIndex`	**Type:** `number` The z-index of the layer. Only applies to Vector and KML layers.

MapsEngineLayerProperties object specification

`google.maps.visualization.MapsEngineLayerProperties` object specification

This object defines the properties of a Maps Engine layer.

Library

visualization

Properties	
`name`	**Type:** `string` The name of the layer.

MapsEngineMouseEvent object specification

`google.maps.visualization.MapsEngineMouseEvent` object specification

The properties of a mouse event on a `MapsEngineLayer`.

Library

visualization

Properties	
`featureId`	**Type:** `string`

	The feature ID, guaranteed to be unique within the layer.
`infoWindowHtml`	**Type:** `string` Pre-rendered HTML content, as placed in the infowindow by the default UI.
`latLng`	**Type:** `LatLng` The position at which to anchor an infowindow on the clicked feature.
`pixelOffset`	**Type:** `Size` The offset to apply to an infowindow anchored on the clicked feature.

MapsEngineStatus class

`google.maps.visualization.MapsEngineStatus` class

The status returned by `MapsEngineLayer` when a layer has loaded.

Library

visualization

Constant	
`INVALID_LAYER`	The requested layer is not a valid layer.
`OK`	The layer loaded successfully.
`UNKNOWN_ERROR`	The layer failed to load for an unknown reason.

DynamicMapsEngineLayer class

`google.maps.visualization.DynamicMapsEngineLayer` class

A `DynamicMapsEngineLayer` allows you to display data from Google Maps Engine or the Google Earth Gallery.

This class extends `MVCObject`.

Library

visualization

Constructor	
DynamicMapsEngineLayer(options:*DynamicMapsEngineLayerOptions*)	Creates a new instance of DynamicMapsEngineLayer.

Methods	
getFeatureStyle(featureId:string)	**Return Value:** FeatureStyle Returns the style for the given feature, with which individual style properties can be retrieved or set.
getLayerId()	**Return Value:** string Returns the ID of the Maps Engine layer being displayed, if set.
getLayerKey()	**Return Value:** string Returns the key of the layer to be displayed.
getMap()	**Return Value:** Map Returns the map on which this layer is displayed.
getMapId()	**Return Value:** string Returns the ID of the Maps Engine map to which the layer belongs.
getOpacity()	**Return Value:** number Returns the opacity of the layer. Applies only to imagery layers.
getStatus()	**Return Value:** MapsEngineStatus Returns the status of the layer, set once the requested layer has loaded.
setLayerId(layerId:string)	**Return Value:** None Sets the ID of a single Maps Engine layer to display.
setLayerKey(layerKey:string)	**Return Value:** None Sets the key of the layer to be displayed. Maps Engine

	Sets the key of the layer to be displayed. Maps Engine Layer Keys are only unique within a single map, and can be changed by map owners. Changing this value will cause the layer to be redrawn.
`setMap(map:Map)`	**Return Value:** None Renders the layer on the specified map. If map is set to null, the layer will be removed.
`setMapId(mapId:string)`	**Return Value:** None Sets the ID of the Maps Engine map to which the layer belongs. Changing this value will cause the layer to be redrawn.
`setOpacity(opacity:number)`	**Return Value:** None Sets the opacity of the layer, expressed as a number between 0 and 1. Applies only to imagery layers. Note: Be careful of setting this option for other layer types, as it may become effective in the future.
`setOptions(options:`*`DynamicMapsEngineLayerOptions`*`)`	**Return Value:** None

Events	
`click`	**Arguments:** *`DynamicMapsEngineMouseEvent`* This event is fired when a feature in the layer is clicked.
`dblclick`	**Arguments:** *`DynamicMapsEngineMouseEvent`* This event is fired when a feature in the layer is double clicked.
`mousedown`	**Arguments:** *`DynamicMapsEngineMouseEvent`* This event is fired for a mousedown on a feature in the layer.
`mousemove`	**Arguments:** *`DynamicMapsEngineMouseEvent`* This event is fired when the mouse moves over a feature in the layer.
`mouseout`	**Arguments:** *`DynamicMapsEngineMouseEvent`* This event is fired when the mouse leaves a feature in the layer.
`mouseover`	**Arguments:** *`DynamicMapsEngineMouseEvent`* This event is fired when the mouse enters a feature in the layer.

mouseup	**Arguments:** *DynamicMapsEngineMouseEvent* This event is fired for a mouseup on a feature in the layer.
properties_changed	**Arguments:** None This event is fired when the layer's properties are available for reading.
rightclick	**Arguments:** *DynamicMapsEngineMouseEvent* This event is fired for a rightclick on a feature in the layer.
status_changed	**Arguments:** None This event is fired when the layer has finished loading, and the status is available to determine if the layer loaded successfully.

DynamicMapsEngineLayerOptions object specification

`google.maps.visualization.DynamicMapsEngineLayerOptions` object specification

This object defines the properties that can be set on a `DynamicMapsEngineLayer` object. `layerId`, or both `mapId` and `layerKey` must be set.

Library

visualization

Properties	
accessToken	**Type:** `string` The authentication token returned by an OAuth 2.0 authentication request.
clickable	**Type:** `boolean` If true, the layer receives mouse events. Default value is true.
layerId	**Type:** `string` The ID of the Maps Engine layer to display.
layerKey	**Type:** `string` The key of the layer to display from the specified map.

`map`	**Type:** `Map` The map on which to display the layer.
`mapId`	**Type:** `string` The ID of the Maps Engine map that contains the layer with the given `layerKey`.
`opacity`	**Type:** `number` The opacity of the layer, expressed as a number between 0 and 1. Defaults to 1. Applies only to imagery layers. Note: Be careful of setting this option for other layer types, as it may become effective in the future.
`suppressInfoWindows`	**Type:** `boolean` Suppress the rendering of info windows when layer features are clicked.

DynamicMapsEngineMouseEvent object specification

`google.maps.visualization.DynamicMapsEngineMouseEvent` object specification

The properties of a mouse event on a `DynamicMapsEngineLayer`.

Library

visualization

Methods	
`getDetails(callback:function(MapsEngineMouseEvent))`	**Return Value:** None Takes a callback that will be called with details about the feature that may be used to render an info window.

Properties	
`featureId`	**Type:** `string` The feature ID, guaranteed to be unique within the layer.
`latLng`	**Type:** `LatLng` The latitude/longitude that was below the cursor when the event occurred.

FeatureStyle object specification

`google.maps.visualization.FeatureStyle` object specification

Library

visualization

Methods	
`reset(property:string)`	**Return Value:** None Resets the given style property to its original value.
`resetAll()`	**Return Value:** None Resets all style properties to their original values.

Properties	
`fillColor`	**Type:** `string` The feature's fill color. All CSS3 colors are supported except for extended named colors.
`fillOpacity`	**Type:** `string` Fill opacity, expressed as a decimal between 0 and 1 inclusive. This property may be set as a number, but it will always be returned as a string.
`iconAnchor`	**Type:** `string` The icon's anchor point is the pixel in the source image that is aligned with the point's geographical location, expressed as a whitespace-separated pair of numbers: `x y`. Defaults to the center of the icon.
`iconClip`	**Type:** `string` The rectangular region of the icon's image (in image pixel coordinates) to use, as a whitespace-separated 4-tuple of numbers: `x y width height`. For example, to use a 32x32 icon situated at (0, 64) in a sprite sheet, specify `0 64 32 32`.
`iconImage`	**Type:** `string` The image to render at the point. Currently, only `url(...)` is supported.

`iconOpacity`	**Type:** `string` Icon opacity, expressed as a decimal between 0 and 1 inclusive. This property may be set as a number, but it will always be returned as a string.
`iconSize`	**Type:** `string` Icon size, expressed as a string with two measurements (with pixel or percentage as unit) separated by whitespace.
`strokeColor`	**Type:** `string` The feature's stroke color. All CSS3 colors are supported except for extended named colors.
`strokeOpacity`	**Type:** `string` Stroke opacity, expressed as a decimal between 0 and 1 inclusive. This property may be set as a number, but it will always be returned as a string.
`strokeWidth`	**Type:** `string` Stroke width in pixels. This property may be set as a number, but it will always be returned as a string.
`zIndex`	**Type:** `string` Rendering order. Features with greater zIndex are rendered on top.

HeatmapLayer class

`google.maps.visualization.HeatmapLayer` class

A layer that provides a client-side rendered heatmap, depicting the intensity of data at geographical points.

This class extends `MVCObject`.

Library

visualization

Constructor	
`HeatmapLayer(opts?:`*`HeatmapLayerOptions`*`)`	Creates a new instance of `HeatmapLayer`.

Methods	
`getData()`	**Return Value:** `MVCArray<LatLng|WeightedLocation>` Returns the data points currently displayed by this heatmap.
`getMap()`	**Return Value:** `Map`
`setData(data:MVCArray<LatLng|WeightedLocation>| Array<LatLng|WeightedLocation>)`	**Return Value:** None Sets the data points to be displayed by this heatmap.
`setMap(map:Map)`	**Return Value:** None Renders the heatmap on the specified map. If map is set to null, the heatmap will be removed.
`setOptions(options:`*`HeatmapLayerOptions`*`)`	**Return Value:** None

HeatmapLayerOptions object specification

`google.maps.visualization.HeatmapLayerOptions` object specification

This object defines the properties that can be set on a `HeatmapLayer` object.

Library

visualization

Properties	
`data`	**Type:** `MVCArray<LatLng>` The data points to display. Required.
`dissipating`	**Type:** `boolean` Specifies whether heatmaps dissipate on zoom. By default, the radius of influence of a data point is specified by the radius option only. When dissipating is disabled, the radius

	option is interpreted as a radius at zoom level 0.
`gradient`	**Type:** `Array<string>` The color gradient of the heatmap, specified as an array of CSS color strings. All CSS3 colors are supported except for extended named colors.
`map`	**Type:** `Map` The map on which to display the layer.
`maxIntensity`	**Type:** `number` The maximum intensity of the heatmap. By default, heatmap colors are dynamically scaled according to the greatest concentration of points at any particular pixel on the map. This property allows you to specify a fixed maximum.
`opacity`	**Type:** `number` The opacity of the heatmap, expressed as a number between 0 and 1. Defaults to 0.6.
`radius`	**Type:** `number` The radius of influence for each data point, in pixels.

WeightedLocation object specification

`google.maps.visualization.WeightedLocation` object specification

A data point entry for a heatmap. This is a geographical data point with a weight attribute.

Library

visualization

Properties	
`location`	**Type:** `LatLng` The location of the data point.
`weight`	**Type:** `number` The weighting value of the data point.

上次更新：九月 3, 2015。