GMap 包 API 文档

Runtian Zhai (gmap.runtianz.cn)

GMap 是根据 shipxy.com 提供的 api 开发的一个地图包。相比原生 API，GMap 的使用更加方便。它不需要显式地定义 mapReady 方法就可以自动管理多线程地图加载，这使得对地图的操作更加安全。同时，GMap 还支持在一个页面中放入多个 GMap 并同时对它们进行操作。

使用 GMap

使用 GMap 需要在 html 文档中引用 GMap.js，它会自动添加 GMap 需要的所有 js 文件

使用如下代码即可创建一个 GMap 对象

```
<script src="GMap.js" type="text/javascript"></script>
var map;
window.onload = function () {
/**
 * 在页面的一个元素内显示地图，并返回该地图的实例
 * @param objectId 必选项，用于显示地图的元素的 id
 * @param centerLat 可选项，地图中心点的纬度，范围为-90 到 90，默认为 30，即北纬 30 度
 * @param centerLng 可选项，地图中心点的经度，默认为 120，即东经 120 度
 * @param zoom 可选项，地图缩放级别，取值为 1~18 内的整数，值越大地图显示范围越小，默认为 5
 * @param mapType 可选项，地图的类型，GMap.CMAP 是海图，
 *        GMap.GOOGLEMAP 是地图，GMap.GOOGLESATELLITE 是卫星图，默认为 GMap.GOOGLEMAP
 *        地图类型的三个常量（取值为其中一个）
 *        GMap.CMAP
 *        GMap.GOOGLEMAP
 *        GMap.GOOGLESATELLITE
 */
    map = new GMap.Map(objectId, centerLat, centerLng, zoom, mapType);
};
```

这段代码会直接在 html 文档中 id 为 objectId 的元素内显示地图，并返回一个 Map 实例

Map 实例的方法

以下是对 Map 实例进行操作的方法。

```
/**
 * 切换地图的中心点以及缩放级别
 * @param centerLat 必选项，地图中心点的纬度
 * @param centerLng 必选项，地图中心点的经度
 * @param zoom 可选项，地图的缩放级别，默认为 5
 */
map.setCenter(centerLat, centerLng, zoom);
```

```
/**
 * 切换地图的缩放级别
 * @param zoom 必选项，地图的缩放级别
 */
map.setZoom(zoom);

/**
 * 切换地图类型
 * @param mapType 必选项，地图的类型
 */
map.setMapType(mapType);

/**
 * 返回一个二元数组，表示当前地图中心的经纬度，前一个是纬度，后一个是经度
 * @returns 当前地图中心的经纬度
 */
var ans = map.getCenter();

/**
 * 返回地图的缩放级别
 * @returns 地图的缩放级别
 */
var ans = map.getZoom();

/**
 * 返回地图的类型，取值为三个地图类型常量中的一个
 * @returns 地图的类型
 */
var ans = map.getMapType();

/**
 * 返回一个二元数组，表示地图所显示的区域的像素大小，前一个是宽度值，后一个是高
度值
 * @returns 当前地图显示区域的像素大小
 */
var ans = map.getSize();

/**
 * 返回一个四元数组，表示当前地图所显示的区域的经纬度范围
 * 这个四元数组是[southWestLat, southWestLng, northEastLat, northEastLng]
 * 分别表示了西南角的经纬度以及东北角的经纬度
 * @returns [southWestLat, southWestLng, northEastLat, northEastLng]
 *          southWestLat 地图所显示区域西南角的纬度
 *          southWestLng 地图所显示区域西南角的经度
```

```
 *          northEastLat 地图所显示区域东北角的纬度
 *          northEastLng 地图所显示区域东北角的经度
 */
var ans = map.getBounds();

/**
 * 按像素向东、向南平移地图
 * @param east 地图向东方的平移像素数，可以为负（即向西平移）
 * @param south 地图向南方的平移像素数，可以为负（即向北平移）
 */
map.translate(east, south);

/**
 * 输入一个经纬度，返回此经纬度对应的点在屏幕上的像素坐标
 * 其中，地图的最左上角的坐标是(0,0)
 * @param lat 点的纬度
 * @param lng 点的经度
 * @returns 一个二元数组，表示这个点在屏幕上的像素坐标，前一个是水平方向，后一个是竖直方向
 */
var ans = map.fromLatLngToPoint(lat, lng);

/**
 * 输入一个屏幕上的点的像素坐标，返回这个点对应的经纬度
 * @param x 点的水平像素坐标
 * @param y 点的竖直像素坐标
 * @returns 一个二元数组，表示这个点的经纬度，前一个是纬度，后一个是经度
 */
var ans = map.fromPointToLatLng(x, y);
```

在地图上进行绘制

可以很方便地在 Map 对象上添加点、折线和多边形。所有在 Map 对象上的绘制物都必须有一个自己的 id。这些 id 可以用来对绘制物进行编辑、删除等操作，因此必须两两不同。如果尝试添加一个绘制物且其 id 已经存在，那么一个异常会被抛出。

注意：由于在程序内部__map__这个 id 被认作全局地图的 id，因此请不要使用这个 id。

例如，下面的代码就可以在地图上绘制一个点。

```
var font = new GMap.Font('Verdana', 11, 0xff33cc, true, true, true);
var fill = new GMap.Fill(0x000000, 1.0);
var stroke = new GMap.Stroke(5, 0xff0000, 0.2);
var label = new GMap.Label('test', 0, 0, font, null, fill, stroke);
map.addPoint('point1', 30, 120, label, null, [3, 12], 3, 0, 0);
```

线条和填充

绘制物的两个重要属性是线条和填充的样式。Stroke 对象可以定义一个绘制物的线条属性，

Fill 对象可以定义一个绘制物的填充属性。Stroke 对象和 Fill 对象都有默认值，如果绘制物体时不传入这两个对象的实例，那么默认值将会被使用。可以修改这个默认值。
以下是 Stroke 对象和 Fill 对象的构造方法

```
/**
 * 返回一个线条属性对象实例
 * @param thickness 线条的粗细，单位为像素，如果是 null 则使用默认值
 * @param color 线条的颜色，取值为 0x00000~0xFFFFFF，如果是 null 则使用默认值
 * @param alpha 线条的透明度，取值为 0.0~1.0,1.0 表示不透明，如果是 null 则使用默认值
 */
var stroke = new GMap.Stroke(thickness, color, alpha);

/**
 * 返回一个默认线条属性的复制
 */
var stroke = new GMap.Stroke();

/**
 * 返回一个填充属性对象实例
 * @param color 填充颜色，取值为 0x00000~0xFFFFFF，如果是 null 则使用默认值
 * @param alpha 填充透明度，取值为 0.0~1.0,1.0 表示不透明，如果是 null 则使用默认值
 */
var fill = new GMap.Fill(color, alpha);

/**
 * 返回一个默认填充属性的复制
 */
var fill = new GMap.Fill();
```

用以下方法可以改变这些属性的默认值。注意，改变默认值只对当前页面有效。当页面刷新或者切换页面后这些改变就会失效。

```
/**
 * 改变线条属性的默认值为传入的 Stroke 实例
 * 初始默认值为粗细 1px，颜色黑色(0x000000)，完全不透明(1.0)
 * @param stroke 一个 Stroke 实例
 */
GMap.setDefaultStroke(stroke);

/**
 * 改变填充属性的默认值为传入的 Fill 实例
 * 初始默认值为颜色黑色(0x000000)，完全不透明(1.0)
 * @param fill 一个 Fill 实例
 */
```

```
GMap.setDefaultFill(fill);
```

字体和标签

Label 对象可以附加在任何绘制物上。如果在绘制物体时传入一个 Label 实例，那么该 Label 就会被附加在绘制物实例上。Font 对象可以定义 Label 对象所显示的文本的字体。Font 对象有一个默认值，如果构造 Label 对象时不传入一个 Font 实例，那么默认值将会被使用。可以修改这个默认值。

以下是 Label 对象和 Font 对象的构造方法

```
/**
 * 返回一个字体属性对象实例
 * @param family 字体名称，如"Calibri"或"宋体"，如果是 null 则使用默认值
 * @param size 字体大小，单位为像素，如果是 null 则使用默认值
 * @param color 字体颜色，取值为 0x00000~0xFFFFFF，如果是 null 则使用默认值
 * @param bold 布尔类型，表示是否加粗，true 表示加粗
 * @param italic 布尔类型，表示是否斜体，true 表示斜体
 * @param underline 布尔类型，表示是否有下划线，true 表示有下划线
 */
var font = new GMap.Font(family, size, color, bold, italic, underline);

/**
 * 返回一个默认字体的复制
 */
var font = new GMap.Font();

/**
 * 返回一个标签实例
 * @param text 必选项，标签上要显示的文字
 * @param labelPosX 可选项，标签相对绘制物的水平偏移量，以向右为正方向，默认为
0
 * @param labelPosY 可选项，标签相对绘制物的竖直偏移量，以向下为正方向，默认为
0
 * @param font 可选项，一个 Font 实例，表示标签中文字的字体，默认为默认字体
 * @param href 可选项，标签中文字的超链接地址，如果为 null 则没有超链接，默认为
null
 * @param backgroundFill 可选项，一个 Fill 实例，表示标签的背景填充样式，如果
为 null 则无背景，默认为 null
 * @param borderStroke 可选项，一个 Stroke 实例，表示标签的边框线条样式，如果
为 null 则无边框，默认为 null
 */
var label = new GMap.Label(text, labelPosX, labelPosY, font, href,
backgroundFill, borderStroke);
```

用以下方法可以改变字体属性的默认值。注意，改变默认值只对当前页面有效。当页面刷新或者切换页面后这些改变就会失效。

```
/**
 * 改变字体属性的默认值为传入的 Font 实例
 * 初始默认值为 Verdana, 11px, 黑色(0x000000), 非粗体，非斜体，没有下划线
 * @param font 一个 Font 实例
 */
GMap.setDefaultFont(font);
```

绘制点

可以用 addPoint 方法在地图上绘制一个点。一个点内绘制的可以是任何一个图片文件。默认情况下绘制采用默认 URL 中的图片。这个默认的 URL 可以修改。

```
/**
 * 在地图上绘制一个点
 * @param id 必选项，点的 id，一个字符串
 *           如果该 id 已经存在，则会抛出一个 GMap.IdAlreadyExistError 类型的异常
 *           如果传入 null，程序会分配一个随机字符串作为该绘制物的 id
 * @param lat 必选项，点的纬度
 * @param lng 必选项，点的经度
 * @param label 可选项，一个 Label 实例，用于在点上显示标签，如为 null 则不显示标签，默认为 null
 * @param imageUrl 可选项，点的图片的 URL，如果不选或为 null 则默认为默认值
 * @param zoomlevels 可选项，一个二元数组[low, high]，只有当地图的缩放级别在区间
 *                   [low, high]之内时这个点才会显示，默认为[1,18]
 * @param zIndex 可选项，绘制物的堆叠层级，默认为 3
 * @param imagePosX 可选项，绘制的点相对于指定点的水平偏移量，以向右为正方向，默认为 0
 * @param imagePosY 可选项，绘制的点相对于指定点的竖直偏移量，以向下为正方向，默认为 0
 * @param isEditable 可选项，表示点是否可以拖动，true 表示可以拖动，默认为 false
 * @returns 点的 id
 */
map.addPoint(id, lat, lng, label, imageUrl, zoomlevels, zIndex, imagePosX, imagePosY, isEditable);
```

可以用以下方法来改变绘制点时采用的默认图片的 URL。注意，改变默认值只对当前页面有效。当页面刷新或者切换页面后这些改变就会失效。

```
/**
 * 改变绘制点时采用的默认图片的 url
 * @param url 一个字符串，新的 url
 */
GMap.setDefaultPointUrl(url);
```

绘制折线

可以用 addPolyline 方法绘制一条折线。

```
/**
 * 在地图上绘制一条折线
 * @param id 必选项，折线的 id，一个字符串
 *           如果该 id 已经存在，则会抛出一个 GMap.IdAlreadyExistError 类型的异常
 *           如果传入 null，程序会分配一个随机字符串作为该绘制物的 id
 * @param data 必选项，一个数组，其中每一个元素都是一个二元数组[lat, lng]，表示一个节点的信息
 *             lat 节点的纬度
 *             lng 节点的经度
 * @param stroke 可选项，线条的样式，一个 Stroke 实例，如为 null 则采用默认线条样式，默认为 null
 * @param label 可选项，一个 Label 实例，用于在折线上显示标签，如为 null 则不显示标签，默认为 null
 * @param zoomlevels 可选项，一个二元数组[low, high]，只有当地图的缩放级别在区间
 *                   [low, high]之内时这条折线才会显示，默认为[1,18]
 * @param zIndex 可选项，绘制物的堆叠层级，默认为 3
 * @param isEditable 可选项，表示折线是否可以拖动，true 表示可以拖动，默认为 false
 * @returns 折线的 id
 */
map.addPolyline(id, data, stroke, label, zoomlevels, zIndex, isEditable);
```

绘制多边形

可以用 addPolygon 方法绘制一个多边形。

```
/**
 * 在地图上绘制一个多边形
 * @param id 必选项，多边形的 id，一个字符串
 *           如果该 id 已经存在，则会抛出一个 GMap.IdAlreadyExistError 类型的异常
 *           如果传入 null，程序会分配一个随机字符串作为该绘制物的 id
 * @param data 必选项，一个数组，其中每一个元素都是一个二元数组[lat, lng]，表示一个顶点的信息
 *             lat 节点的纬度
 *             lng 节点的经度
 * @param stroke 可选项，多边形的边的线条样式，一个 Stroke 实例，如为 null 则采用默认线条样式，默认为 null
 * @param fill 可选项，多边形的填充样式，一个 Fill 实例，如为 null 则采用默认填充样式，默认为 null
 * @param label 可选项，一个 Label 实例，用于在多边形上显示标签，如为 null 则不显示标签，默认为 null
 * @param zoomlevels 可选项，一个二元数组[low, high]，只有当地图的缩放级别在
```

```
区间
 *                  [low, high]之内时这个多边形才会显示，默认为[1,18]
 * @param zIndex 可选项，绘制物的堆叠层级，默认为 3
 * @param isEditable 可选项，表示多边形是否可以拖动，true 表示可以拖动，默认为
false
 * @returns 多边形的 id
 */
 map.addPolygon(id, data, stroke, fill, label, zoomlevels, zIndex,
isEditable);
```

控制绘制物

你可以通过在绘制时已经设定的 id 方便地对地图上的绘制物进行控制。你可以通过 id 得到该绘制物的类型，取值为三个类型常量：GMap.POINT, GMap.POLYLINE, GMap.POLYGON。你还可以通过 id 隐藏、显示或删除一个绘制物。如果你输入了一个不存在的 id，那么程序会抛出一个 GMap.IdNotExistError 类型的异常。

以下是相关的一些方法：

```
/**
 * 返回一个绘制物的类型，取值为三个类型常量之一
 * @param id 绘制物的 id
 * @returns 绘制物的类型，如果 id 不存在则抛出 GMap.IdNotExistError 类型的异常
 */
var ans = map.getType(id);

/**
 * 返回地图上存在的，同一个类型的全部绘制物
 * @param type 绘制物的类型，取值为三个类型常量之一
 * @returns 一个数组，包含所有此类绘制物的 id
 */
var ans = map.getByType(type);

/**
 * 移动地图，使得某一个绘制物成为中心点，并改变地图的缩放级别
 * @param id 必选项，绘制物的 id，如果 id 不存在则抛出 GMap.IdNotExistError 类
型的异常
 * @param zoom 可选项，地图新的缩放级别，如果不选，如果是点类绘制物则不改变缩
放级别，
 *             如果是折线或者多边形，则会按照其范围自动修改到合适的缩放级别
 */
map.locate(id, zoom);

/**
 * 删除一个绘制物，可以删除已隐藏的绘制物
 * @param id 要删除的绘制物的 id，如果 id 不存在则抛出 GMap.IdNotExistError 类
```

```
型的异常
 */
map.remove(id);

/**
 * 删除地图上存在的，某一个类型的全部绘制物，包括已经隐藏的绘制物
 * @param type 绘制物的类型，取值为三个类型常量之一
 * @returns 一个整数，成功删除的绘制物的个数
 */
map.removeByType(type);

/**
 * 删除地图上存在的所有绘制物，包括已经隐藏的绘制物
 * @returns 一个整数，成功删除的绘制物的个数
 */
map.removeAll();

/**
 * 隐藏一个绘制物并返回是否成功，如果该绘制物已经隐藏则视为不成功，但不会抛出异
常
 * @param id 要隐藏的绘制物的 id，如果 id 不存在则抛出 GMap.IdNotExistError 类
型的异常
 * @returns 一个布尔值，是否成功隐藏该绘制物
 */
map.hide(id);

/**
 * 隐藏地图上某一个类型的全部绘制物，返回成功隐藏的绘制物的个数
 * @param type 绘制物的类型，取值为三个类型常量之一
 * @returns 被隐藏的绘制物的个数，不包括原先就已经隐藏了的
 */
map.hideByType(type);

/**
 * 隐藏地图上的全部绘制物，返回成功隐藏的绘制物的个数
 * @returns 被隐藏的绘制物的个数，不包括原先就已经隐藏了的
 */
map.hideAll();

/**
 * 显示一个被隐藏的绘制物，返回是否成功显示，如果该绘制物没有被隐藏则视为不成功
 * 但不会抛出异常
 * @param id 要显示的绘制物的 id，如果 id 不存在则抛出 GMap.IdNotExistError 类
型的异常
```

```
 * @returns 一个布尔值，绘制物是否成功显示
 */
map.show(id);

/**
 * 在地图上显示全部被隐藏的某一个类型的绘制物，返回被显示的绘制物的数量
 * @param type 绘制物的类型，取值为三个类型常量之一
 * @returns 一个整数，被显示的绘制物的数量，不包括已经显示在地图上的
 */
map.showByType(type);

/**
 * 在地图上显示所有被隐藏的绘制物，返回被显示的绘制物的数量
 * @returns 一个整数，被显示的绘制物的数量，不包括已经显示在地图上的
 */
map.showAll();

/**
 * 返回某一个绘制物是否被隐藏了
 * @param id 绘制物的 id，如果 id 不存在则抛出 GMap.IdNotExistError 类型的异常
 * @returns 一个布尔值，true 表示该绘制物被隐藏
 */
var ans = map.isHidden(id);
```

事件

通过在地图上注册事件监听器，可以达到与用户互动的目的。GMap 中一共有两种事件，地图事件——包括地图的拖动、缩放、改变地图类型等，绘制物事件——包括点击、拖动、增加/删除、隐藏/显示绘制物等。地图事件的主体是整个地图，而绘制物事件的主体是单个事件。

要在地图上应用事件，首先要给地图或地图上的某个绘制物注册事件监听器。

```
/**
 * 为当前地图注册一系列事件监听器
 * @param events 一个字典，保存了一组事件监听器
 */
map.addMapEvent(events);

/**
 * 为某一个绘制物注册一系列事件监听器
 * @param id 绘制物的 id，如果 id 不存在则抛出 GMap.IdNotExistError 类型的异常
 * @param events 一个字典，保存了一组事件监听器
 */
map.addOverlayEvent(id, events);
```

```
/**
 * 为某一个类型的全部绘制物注册一系列事件监听器
 * @param type 绘制物的类型，取值为三个类型常量之一
 * @param events 一个字典，保存了一组事件监听器
 */
map.addOverlayEventByType(type, events);

/**
 * 为所有绘制物注册事件监听器
 * @param events 一个字典，保存了一组事件监听器
 */
map.addOverlayEventToAll(events);
```

关于删除绘制物的注释：每当一个绘制物被删除，其上的所有事件会被自动删除。

编写事件字典

注册事件监听器时，需要把所需要的事件记录在事件字典中。例如如下示例：

```
map.addOverlayEvent('point1', {
    'onclick': function(event) { /*...*/ },
    'onmouseover': function(event) { /*...*/ }
});
```

这段代码为 id 为 point1 的一个绘制物注册了两个事件监听器，一个会在绘制物被点击时触发，还有一个会在鼠标移动到绘制物上方时触发。

以下是事件字典的参考表：

地图事件一览表

事件类型	事件触发条件
onchange	地图发生任何变化，包括地图的拖动及缩放
onzoom	地图被缩放
onchangemaptype	地图的类型被改变
onadd	地图上放入了一个新的绘制物
onremove	地图上的一个绘制物被删除
onhide	地图上的一个绘制物被隐藏（已经隐藏的再被隐藏不会触发）
onshow	地图上的一个绘制物被显示（未被隐藏的再被显示不会触发）
onclick	地图被点击
ondoubleclick	地图被双击
onmousedown	地图上的鼠标被摁下
onmouseup	地图上的鼠标被抬起
onmousemove	地图上的鼠标移动
onmouseover	鼠标移入地图
onmouseout	鼠标移出地图

绘制物事件一览表

事件类型	事件触发条件

onremove	绘制物被移除后
onhide	绘制物被隐藏后（若原来已经隐藏则不触发）
onshow	绘制物被显示后（若原来未被隐藏则不触发）
onclick	绘制物被点击
ondoubleclick	绘制物被双击
onmousedown	绘制物上的鼠标被摁下
onmouseup	绘制物上的鼠标被抬起
onmousemove	绘制物上的鼠标移动
onmouseover	鼠标移入绘制物
onmouseout	鼠标移出绘制物

编辑事件函数

当一个事件被触发时，会有一个字典 event 传给事件函数。以下是这个字典的参考表：

event['lat']	事件中心点的纬度
event['lng']	事件中心点的经度
event['id']	事件主体的 id，如果是地图事件，id 则为__map__

事件的中心点可能是鼠标的位置也可能是地图的中心点。所有鼠标事件的中心点都是鼠标的位置，其它事件的中心点都是地图的中心点。

下面会通过一些示例来具体阐释事件函数的编写。

```
map.addOverlayEvent('point1', {  // 示例 1
    'onclick': function(event) {
        alert(event['lat'] + 'and' + event['lng'] + 'and' + event['id']);
    }
});
```

示例 1 中，每当用户点击 point1 这个绘制物，就会弹出一个消息告诉用户鼠标点击位置的经纬度以及 point1 的 id，也就是 point1。

```
map.addMapEvent({  // 示例 2
    'onmouseover': function(event) {
        alert('You are in!');
    }
});
```

示例 2 中，每当用户把鼠标移入地图中就会弹出一条消息。

```
map.addMapEvent({  // 示例 3
    'onhide': function(event) {
        alert('You hide something! It is ' + event['id']);
    }
});
```

示例 3 的意图是，每当一个绘制物被隐藏，就会弹出一条消息告知用户该绘制物的 id。示例 3 的代码的确可以做到每当一个绘制物被隐藏就弹出一条消息，但消息中包含的 id 始终是__map__，这是因为 event['id']存放的是事件主体，而地图事件的主体永远是地图自身。

要达到这一效果应该把事件加给全部的绘制物，如示例 4 所示。

```
map.addOverlayEventToAll({  // 示例 4
    'onhide': function(event) {
        alert('You hide something! It is ' + event['id']);
    }
});
```

事件的移除

可以通过以下方法移除事件。

```
/**
 * 删除某一类型的全部地图事件
 * @param type 一个字符串，类型名
 */
map.removeMapEvent(type);

/**
 * 删除一个绘制物的某一类型的全部绘制物事件
 * @param id 绘制物的 id，如果 id 不存在则抛出 GMap.IdNotExistError 类型的异常
 * @param type 一个字符串，类型名
 */
map.removeOverlayEvent(id, type);

/**
 * 删除一类绘制物的某一类型的全部绘制物事件
 * @param overlayType 绘制物的类型，取值为三个类型常量之一
 * @param eventType 一个字符串，事件的类型名
 */
map.removeOverlayEventByType(overlayType, eventType);

/**
 * 删除某一类型的全部绘制物事件
 * @param type 一个字符串，事件的类型名
 */
map.removeAllOverlayEvent(type);
```

例如，下面这段代码会把所有的注册在点类绘制物上的鼠标单击事件全部移除。

```
map.removeOverlayEventByType(GMap.POINT, 'onclick');
```

组合绘制物

有些时候用户会希望把一些绘制物当做一个整体来看待。这些绘制物可以同时显示或隐藏，也可以给其加上一个共同的事件。用以下方式来组合一组绘制物或删除一个组合。

```
/**
 * 组合一组绘制物
 * @param id 组合的 id，如为 null 则系统分配一个随机字符串作为组合的 id
```

```
 *          如果该 id 已经存在，则会抛出一个 GMap.IdAlreadyExistError 类型的异
常
 * @param idArray 一个绘制物 id 的数组，包含要组合的绘制物
 *                只要其中任意一个 id 不存在，就会抛出 GMap.IdNotExistError 类
型的异常
 * @returns 一个字符串，组合的 id
 */
map.group(id, idArray);

/**
 * 删除一个绘制物组合，如果这个组合不存在也不会抛出异常，而是以返回值来指示
 * @param id 要删除的组合的 id
 * @returns 一个布尔值，组合原本是否存在，false 表示组合原本不存在
 */
map.ungroup(id);

/**
 * 返回一个绘制物组合中的所有绘制物的 id
 * @param id 组合的 id，如果该 id 不存在则抛出 GMap.IdNotExistError 类型的异常
 * @returns 一个数组，包含组合中所有绘制物的 id
 */
var ans = map.getGroup(id);
```

关于删除绘制物的注释：每当删除一个绘制物时，这个绘制物也会同时从它所属的所有组合中被删去。但需要注意的是，即便一个组合内的所有绘制物都被删除，这个组合本身仍然存在。只有调用 ungroup 方法才能够销毁组合本身。当然，也可以调用 removeGroup 方法，同时销毁一个组合以及组合内的所有绘制物。

```
/**
 * 移除一个组合内的所有绘制物，然后删除这个组合
 * @param id 组合的 id，如果该 id 不存在则抛出 GMap.IdNotExistError 类型的异常
 */
map.removeGroup(id);

/**
 * 隐藏一个组合内的所有绘制物
 * @param id 组合的 id，如果该 id 不存在则抛出 GMap.IdNotExistError 类型的异常
 * @returns 一个整数，表示成功被隐藏的绘制物的个数
 */
map.hideGroup(id);

/**
 * 显示一个组合内的所有绘制物
 * @param id 组合的 id，如果该 id 不存在则抛出 GMap.IdNotExistError 类型的异常
```

```
 * @returns 一个整数，表示成功被显示的绘制物的个数
 */
map.showGroup(id);

/**
 * 定位一个组合内的绘制物，系统首先计算出一个囊括整个组合的大矩形，然后把地图中心点设为矩形中心
 * 这里包括被隐藏的绘制物
 * @param id 组合的 id，如果该 id 不存在则抛出 GMap.IdNotExistError 类型的异常
 * @param zoom 可选项，定位后显示的缩放级别，如果不选则由系统自动计算一个合适的缩放级别
 */
map.locateGroup(id, zoom);
```

组合事件

如果说组合绘制物仅仅是提供了一种整体看待绘制物的方式，可能其重要性并没有那么大。组合绘制物的关键在于可以编写组合事件。组合事件的意义在于，当某个操作使得一组事件监听器同时被触发时，属于同一个组合的绘制物的组合事件监听器只会被触发一次。

组合事件的编写方法和普通事件完全一样：

```
/**
 * 为一个绘制物组合注册一系列组合事件监听器
 * @param id 组合的 id，如果该 id 不存在则抛出 GMap.IdNotExistError 类型的异常
 * @param events 一个字典，保存了一组事件监听器
 */
map.addGroupEvent('group1', events);

/**
 * 删除一个绘制物组合的某一类型的全部事件
 * @param id 组合的 id，如果该 id 不存在则抛出 GMap.IdNotExistError 类型的异常
 * @param type 一个字符串，事件的类型名称
 */
map.removeGroupEvent(id, type);
```

需要注意的是组合事件的主体是组合本身，因此 event['id']将会是组合的 id。

我们为什么需要组合事件？考虑一种情况，我们希望每当地图上有点被隐藏时，就弹出一个窗口提示我们。一种方法是给所有点类绘制物注册事件监听器。

```
var map;
window.onload = function () {
    map = new GMap.Map('mapDiv');
    map.addPoint('point1', 29, 115, new GMap.Label('point1'), null, [1,18], 3, 0, 0);
    map.addPoint('point2', 25, 130, new GMap.Label('point2'), null, [1,18], 3, 0, 0);
    map.addPolyline('line', [[29,115], [25,130]], new GMap.Stroke(20,
```

```
0x000000, 1.0));
    map.addOverlayEventByType(GMap.POINT, {
        'onhide': function() {
            alert('You hide a point!');
        }
    });
    map.hideByType(GMap.POINT);
};
```

执行这段代码，我们发现一共连续弹出了两次窗口，因为每一个点被隐藏都会弹出一个窗口，而这不是我们的本意。我们希望如果若干个点是同时被隐藏的，那么只会弹出一次窗口。使用组合事件就可以很容易实现这一点。

```
var map;
window.onload = function () {
    map = new GMap.Map('mapDiv');
    map.addPoint('point1', 29, 115, new GMap.Label('point1'), null, [1,18],
3, 0, 0);
    map.addPoint('point2', 25, 130, new GMap.Label('point2'), null, [1,18],
3, 0, 0);
    map.addPolyline('line', [[29,115], [25,130]], new GMap.Stroke(20,
0x000000, 1.0));
    map.group('points', map.getByType(GMap.POINT));
    map.addGroupEvent('points', {
        'onhide': function() {
            alert('You hide a point!');
        }
    });
    map.hideByType(GMap.POINT);
};
```

这里把所有的点放入一个 id 为 points 的组合里，并给这个组合添加一个组合事件。此时就只会弹出一次窗口。

在另一种情形下，我们手动地删除几个点，但同时仍然希望最终只弹出一个提示窗口。

```
var map;
window.onload = function () {
    map = new GMap.Map('mapDiv');
    map.addPoint('point1', 29, 115, new GMap.Label('point1'), null, [1,18],
3, 0, 0);
    map.addPoint('point2', 25, 130, new GMap.Label('point2'), null, [1,18],
3, 0, 0);
    map.addPolyline('line', [[29,115], [25,130]], new GMap.Stroke(20,
0x000000, 1.0));
    map.group('points', map.getByType(GMap.POINT));
    map.addGroupEvent('points', {
```

```
        'onhide': function() {
            alert('You hide a point!');
        }
    });
    map.hide('point1');
    map.hide('point2');
};
```

运行这段代码会弹出两个窗口，因为程序不认为两次隐藏是同时发生的。如果要让程序认为一系列操作是同时进行的，可以使用 Map 类的 run 方法，如下所示。

```
/**
 * 运行一个函数，函数内的所有代码对组合事件视为同时执行
 * @param fun 一个函数
 * @returns 函数的运行结果
 */
map.run(fun);
```

例如运行下面这段代码：

```
var map;
window.onload = function () {
    map = new GMap.Map('mapDiv');
    map.addPoint('point1', 29, 115, new GMap.Label('point1'), null, [1,18],
3, 0, 0);
    map.addPoint('point2', 25, 130, new GMap.Label('point2'), null, [1,18],
3, 0, 0);
    map.addPolyline('line', [[29,115], [25,130]], new GMap.Stroke(20,
0x000000, 1.0));
    map.group('points', map.getByType(GMap.POINT));
    map.addGroupEvent('points', {
        'onhide': function() {
            alert('You hide a point!');
        }
    });
    map.run(function() {
        map.hide('point1');
        map.hide('point2');
    });
};
```

此时提示窗口只会弹出一次。

线程安全和地图锁

因为地图的加载可能会花费较长的时间（特别是网络不佳的时候），因此地图的加载不在主线程中执行。此时，用户对地图的所有操作都会暂时存放在等待队列中，等地图加载完毕后，

程序逐一执行等待队列中的操作。
地图锁针对的是这样一种情形：用户在地图加载完毕之前，对地图进行了过多的操作（例如用户在等待过程中不断地点击某一个按钮），当地图加载完成时，系统逐一执行所有操作，可能会导致程序一时无法响应。
为了避免这样的情形，确保在地图加载完成前不会进行任何操作，可以像这样打开地图锁。

```
/**
 * 锁上地图锁
 */
GMap.lock();
```

默认情况下地图锁是关闭的，并且地图锁一旦打开就不能再次关闭。地图锁是全局的，即同一个页面上只有一个地图锁。
例如：

```
var map;
window.onload = function () {
    map = new GMap.Map('mapDiv');
    map.addPoint('point1', 29, 115, null, [1,18], 3, 0, 0, new GMap.Label('point1'));
    map.addPoint('point2', 25, 130, null, [1,18], 3, 0, 0, new GMap.Label('point2'));
};
```

执行这段代码后，在地图尚未加载完成前，两个向地图添加点的指令会储存在等待队列中，等地图加载完毕后再执行。因此，地图显示时，上面就会有两个点。
但如果打开了地图锁：

```
var map;
window.onload = function () {
    map = new GMap.Map('mapDiv');
    GMap.lock();
    map.addPoint('point1', 29, 115, null, [1,18], 3, 0, 0, new GMap.Label('point1'));
    map.addPoint('point2', 25, 130, null, [1,18], 3, 0, 0, new GMap.Label('point2'));
};
```

此时，直到地图加载完成前的所有操作都是无效的（它们永远不会被执行），因此执行后地图上并没有点。
但有时候，开发人员可能希望在地图出现时已经出现了一些绘制物，却又不希望用户在地图加载完成前进行任何操作。此时，可以使用 GMap.onready 属性。这是一个全局属性，表示在一个页面上的所有地图都加载完成的瞬间需要执行的操作。这个属性会忽视地图锁。
因此对于这段代码而言：

```
var map;
window.onload = function () {
    map = new GMap.Map('mapDiv');
    GMap.lock();
    map.addPoint('point1', 29, 115, null, [1,18], 3, 0, 0, new GMap.Label('point1'));
    GMap.onready = function() {
        map.addPoint('point2', 25, 130, null, [1,18], 3, 0, 0, new
```

```
GMap.Label('point2'));
    };
};
```

地图加载完毕后，point1 没有显示，事实上 point1 根本没有被构造，而 point2 会显示。