百度地图 API1.3开发指南

API 1.3 2012-5-11

简介

什么是百度地图 API？

百度地图 API 是一套由 JavaScript 语言编写的应用程序接口，它能够帮助您在网站中构建功能丰富、交互性强的地图应用。百度地图 API 包含了构建地图基本功能的各种接口，提供了诸如本地搜索、路线规划等数据服务。

面向的读者

API 是提供给那些具有一定 JavaScript 编程经验和了解面向对象概念的读者使用。此外，读者还应该对地图产品有一定的了解。

您在使用中遇到任何问题，都可以通过 API 贴吧或交流群反馈给我们。

获取 API

地图 API 是由 JavaScript 语言编写的，您在使用之前需要通过<script>标签将 API 引用到页面中：

```
<script src="http://api.map.baidu.com/api?v=1.3" type="text/javascript"></script>
```

其中参数 v 为 API 当前的版本号，目前最新版本为1.3。在1.2版本之前您还可以设置 services 参数，以告知 API 是否加载服务部分，true 表示加载，false 表示不加载，默认为 true。

坐标转换

国际经纬度坐标标准为 WGS-84,国内必须至少使用国测局制定的 GCJ-02,对地理位置进行首次加密。百度坐标在此基础上，进行了 BD-09二次加密措施,更加保护了个人隐私。百度对外接口的坐标系并不是 GPS 采集的真实经纬度，需要通过坐标转换接口进行转换。

坐标转换示例：http://dev.baidu.com/wiki/static/map/API/examples/?v=1.3&0_6#0&6

批量坐标转换示例：http://dev.baidu.com/wiki/static/map/API/examples/?v=1.3&0_7#0&7

开发移动平台上的地图应用

API 自1.1版本起开始支持 iPhone、Android 这样的移动平台。用户通过手机浏览器就可以访问由地图 API 创建出来的应用。移动平台的屏幕尺寸通常比 PC 或笔记本要小，操作方式也有所不同。为了更好的在手机浏览器上展示地图，我们有如下建议：

- 将地图容器高设置为100%，使其充满整个屏幕，或者您也可以计算浏览器窗口的大小并进行设置。
- 添加下面的 meta 标签：<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />

这样做是为了让页面以正常比例进行显示并且禁止用户缩放页面的操作。

您可以参考 Apple's Developer documentation 和 Android documentation 获得更多信息。

异步加载

API 1.1,1.2和1.3版本支持异步加载，您可以在引用脚本的时候添加 callback 参数，当脚本加载完成后 callback 函数会被立刻调用。请参考下面的使用示例：

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>
<title>异步加载</title>
<script type="text/javascript">
function initialize() {
var mp = new BMap.Map('map');
mp.centerAndZoom(new BMap.Point(121.491, 31.233), 11);
}

function loadScript() {
var script = document.createElement("script");
script.src = "http://api.map.baidu.com/api?v=1.2&callback=initialize";
document.body.appendChild(script);
}

window.onload = loadScript;
</script>
</head>
<body>
<div id="map" style="width:500px;height:320px"></div>
</body>
</html>
```

兼容性

浏览器：IE 6.0+、Firefox 3.6+、Opera 9.0+、Safari 3.0+、Chrome

操作系统：Windows、Mac、Linux

移动平台：iPhone、Android

版本说明

地址 http://api.map.baidu.com/api?v=1.3 中的参数 v 表示您加载 API 的版本，例如当前 API 的最新版本为1.3，则您可在地址中添加 v=1.3。当 API 升级后，如果已有接口在使用、命名等方面发生了变化，我们会为其增加一个新的版本号，这不会对您现有的应用造成任何影响。如果升级只是修复一些 bug 或者在不影响现有功能的前提下增加接口、改善性能，则版本号不会发生变化。您可以在更新日志页面查看版本的变化。

问题解答

如果您在使用百度地图 API 中遇到问题，请尝试通过以下途径解决：

- 确认您使用了正确的地图 API 地址。
- 访问百度地图 API 吧，查找相关问题的帖子，或者将您的问题发布到贴吧中。
- 常看《常见问题》指南 http://dev.baidu.com/wiki/map/index.php?title=%E5%B8%B8%E8%A7%81%E9%97%AE%E9%A2%98
- 加入百度 HI 群

基础知识

百度地图的“Hello, World”

开始学习百度地图 API 最简单的方式是看一个简单的示例。以下代码创建了一个地图并以天安门作为地图的中心：

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Hello, World</title>
<style type="text/css">
html{height:100%}
body{height:100%;margin:0px;padding:0px}
#container{height:100%}
</style>
<script                                                    type="text/javascript"
```

```
src="http://api.map.baidu.com/api?v=1.3"></script>
</head>

<body>
<div id="container"></div>
<script type="text/javascript">
var map = new BMap.Map("container"); // 创建地图实例
var point = new BMap.Point(116.404, 39.915); // 创建点坐标
map.centerAndZoom(point, 15); // 初始化地图，设置中心点坐标和地图级别
</script>
</body>
</html>
```

下面我们分步向您介绍：

准备页面

根据 HTML 标准，每一份 HTML 文档都应该声明正确的文档类型，我们建议您使用最新的符合 HTML5规范的文档声明：

```
<!DOCTYPE html>
```

您也可以根据需要选择其他类型的文档声明，这样浏览器会以标准的方式对页面进行渲染，保证页面最大的兼容性。我们不建议您使用 quirks 模式进行开发。

下面我们添加一个 meta 标签，以便使您的页面更好的在移动平台上展示。

```
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
```

接着我们设置样式，使地图充满整个浏览器窗口：

```
<style type="text/css">
html{height:100%}
body{height:100%;margin:0px;padding:0px}
#container{height:100%}
</style>
```

引用百度地图 API 文件

```
<script                                                    type="text/javascript"
src="http://api.map.baidu.com/api?v=1.2"></script>
```

创建地图容器元素

地图需要一个 HTML 元素作为容器，这样才能展现到页面上。这里我们创建了一个 div 元素。

命名空间

API 使用 BMap 作为命名空间，所有类均在该命名空间之下，比如：BMap.Map、BMap.Control、BMap.Overlay。

创建地图实例

```
var map = new BMap.Map("container");
```

位于 BMap 命名空间下的 Map 类表示地图，通过 new 操作符可以创建一个地图实例。其参数可以是元素 id 也可以是元素对象。

注意在调用此构造函数时应确保容器元素已经添加到地图上。

创建点坐标

```
var point = new BMap.Point(116.404, 39.915);
```

这里我们使用 BMap 命名空间下的 Point 类来创建一个坐标点。Point 类描述了一个地理坐标点，其中116.404表示经度，39.915表示纬度。

地图初始化

```
map.centerAndZoom(point, 15);
```

在创建地图实例后，我们需要对其进行初始化，BMap.Map.centerAndZoom()方法要求设置中心点坐标和地图级别。 地图必须经过初始化才可以执行其他操作。

地图配置与操作

地图被实例化并完成初始化以后，就可以与其进行交互了。API 中的地图对象的外观与行为与百度地图网站上交互的地图非常相似。它支持鼠标拖拽、滚轮缩放、双击放大等交互功能。您也可以修改配置来改变这些功能。 比如，默认情况下地图不支持鼠标滚轮缩放操作，因为这样可能会影响整个页面的用户体验，但是如果您希望在地图中使用鼠标滚轮控制缩放，则可以调用 map.enableScrollWheelZoom 方法来开启。配置选项可以在 Map 类参考的配置方法一节中找到。

此外，您还可以通过编程的方式与地图交互。Map 类提供了若干修改地图状态的方法。例如：setCenter()、panTo()、zoomTo()等等。

下面示例显示一个地图，等待两秒钟后，它会移动到新中心点。panTo()方法将让地图平滑移动至新中心点，如果移动距离超过了当前地图区域大小，则地图会直跳到该点。

```
var map = new BMap.Map("container");
var point = new BMap.Point(116.404, 39.915);
map.centerAndZoom(point, 15);
```

```
window.setTimeout(function(){
map.panTo(new BMap.Point(116.409, 39.918));
}, 2000);
```

控件

地图控件概述

百度地图上负责与地图交互的 UI 元素称为控件。百度地图 API 中提供了丰富的控件，您还可以通过 Control 类来实现自定义控件。

地图 API 中提供的控件有：

Control：控件的**抽象基类**，所有控件均继承此类的方法、属性。通过此类您可实现自定义控件。

NavigationControl：地图**平移缩放控件**，默认位于地图左上方，它包含控制地图的平移和缩放的功能。

OverviewMapControl：**缩略地图控件**，默认位于地图右下方，是一个可折叠的缩略地图。

ScaleControl：**比例尺控件**，默认位于地图左下方，显示地图的比例关系。

MapTypeControl：**地图类型控件**，默认位于地图右上方。

CopyrightControl：**版权控件**，默认位于地图左下方。

向地图添加控件

可以使用 Map.addControl()方法向地图添加控件。在此之前地图需要进行初始化。例如，要将标准地图控件添加到地图中，可在代码中添加如下内容：

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 11);
map.addControl(new BMap.NavigationControl());
```

可以向地图添加多个控件。在本例中我们向地图添加一个平移缩放控件、一个比例尺控件和一个缩略图控件。在地图中添加控件后，它们即刻生效。

```
map.addControl(new BMap.NavigationControl());
map.addControl(new BMap.ScaleControl());
map.addControl(new BMap.OverviewMapControl());
map.addControl(new BMap.MapTypeControl());
map.setCurrentCity("北京"); // 仅当设置城市信息时，MapTypeControl 的切换功能才能可用
```

控制控件位置

初始化控件时，可提供一个可选参数，其中的 anchor 和 offset 属性共同控制控件在地图上的位置。

控件停靠位置

anchor 表示控件的停靠位置，即控件停靠在地图的哪个角。当地图尺寸发生变化时，控件会根据停靠位置的不同来调整自己的位置。anchor 允许的值为：

BMAP_ANCHOR_TOP_LEFT 表示控件定位于地图的左上角。

BMAP_ANCHOR_TOP_RIGHT 表示控件定位于地图的右上角。

BMAP_ANCHOR_BOTTOM_LEFT 表示控件定位于地图的左下角。

BMAP_ANCHOR_BOTTOM_RIGHT 表示控件定位于地图的右下角。

控件位置偏移

除了指定停靠位置外，还可以通过偏移量来指示控件距离地图边界有多少像素。如果两个控件的停靠位置相同，那么控件可能会重叠在一起，这时就可以通过偏移值使二者分开显示。

本示例将比例尺放置在地图的左下角，由于 API 默认会有版权信息，因此需要添加一些偏移值以防止控件重叠。

```
var opts = {offset: new BMap.Size(150, 5)}
map.addControl(new BMap.ScaleControl(opts));
```

修改控件配置

地图 API 的控件提供了丰富的配置参数，您可参考 API 文档来修改它们以便得到符合要求的控件外观。例如，NavigationControl 控件就提供了如下类型：

BMAP_NAVIGATION_CONTROL_LARGE 表示显示完整的平移缩放控件。

BMAP_NAVIGATION_CONTROL_SMALL 表示显示小型的平移缩放控件。

BMAP_NAVIGATION_CONTROL_PAN 表示只显示控件的平移部分功能。

BMAP_NAVIGATION_CONTROL_ZOOM 表示只显示控件的缩放部分功能。

下图从左向右依次展示了上述不同类型的控件外观：

下面的示例将调整平移缩放地图控件的外观。

```
var opts = {type: BMAP_NAVIGATION_CONTROL_SMALL}
map.addControl(new BMap.NavigationControl(opts));
```

自定义控件

百度地图 API 允许您通过继承 Control 来创建自定义地图控件。

要创建可用的自定义控件，您需要做以下工作：

定义一个自定义控件的构造函数。

设置自定义控件构造函数的 prototype 属性为 Control 的实例，以便继承控件基类。

实现 initialize()方法并提供 defaultAnchor 和 defaultOffset 属性。

定义构造函数并继承 Control

首先您需要定义自定义控件的构造函数，并在构造函数中提供 defaultAnchor 和 defaultOffset 两个属性，以便 API 正确定位控件位置，接着让其继承于 Control。在下面的示例中我们定义一个名为 ZoomControl 的控件，每一次点击将地图放大两个级别。它具有文本标识，而不是平移缩放控件中使用的图形图标。

```
// 定义一个控件类，即 function
function ZoomControl(){
 // 设置默认停靠位置和偏移量
 this.defaultAnchor = BMAP_ANCHOR_TOP_LEFT;
 this.defaultOffset = new BMap.Size(10, 10);
}
// 通过 JavaScript 的 prototype 属性继承于 BMap.Control
ZoomControl.prototype = new BMap.Control();
```

初始化自定义控件

当调用 map.addControl()方法添加自定义控件时，API 会调用该对象的 initialize()方法用来初始化控件，您需要实现此方法并在其中创建控件所需的 DOM 元素，并添加 DOM 事件。所有自定义控件中的 DOM 元素最终都应该添加到地图容器（即地图所在的 DOM 元素）中去，地图容器可以通过 map.getContainer()方法获得。最后 initialize()方法需要返回控件容器的 DOM 元素。

```
// 自定义控件必须实现 initialize 方法，并且将控件的 DOM 元素返回
// 在本方法中创建个 div 元素作为控件的容器，并将其添加到地图容器中
ZoomControl.prototype.initialize = function(map){
 // 创建一个 DOM 元素
 var div = document.createElement("div");
 // 添加文字说明
 div.appendChild(document.createTextNode("放大2级"));
 // 设置样式
 div.style.cursor = "pointer";
 div.style.border = "1px solid gray";
 div.style.backgroundColor = "white";
 // 绑定事件，点击一次放大两级
 div.onclick = function(e){
   map.zoomTo(map.getZoom() + 2);
 }
 // 添加 DOM 元素到地图中
 map.getContainer().appendChild(div);
 // 将 DOM 元素返回
 return div;
}
```

添加自定义控件

添加自定义控件与添加其他控件方法一致，调用 map.addControl()方法即可。

```
// 创建控件实例
var myZoomCtrl = new ZoomControl();
// 添加到地图当中
map.addControl(myZoomCtrl);
```

覆盖物

地图覆盖物概述

所有叠加或覆盖到地图的内容，我们统称为地图覆盖物。如标注、矢量图形元素(包括：折线和多边形和圆)、信息窗口等。覆盖物拥有自己的地理坐标，当您拖动或缩放地图时，它们会相应的移动。

地图 API 提供了如下几种覆盖物：

Overlay：覆盖物的抽象基类，所有的覆盖物均继承此类的方法。

Marker：标注表示地图上的点，可自定义标注的图标。

Label：表示地图上的文本标注，您可以自定义标注的文本内容。

Polyline：表示地图上的折线。

Polygon：表示地图上的多边形。多边形类似于闭合的折线，另外您也可以为其添加填充颜色。

Circle: 表示地图上的圆。

InfoWindow：信息窗口也是一种特殊的覆盖物，它可以展示更为丰富的文字和多媒体信息。注意：同一时刻只能有一个信息窗口在地图上打开。

可以使用 map.addOverlay 方法向地图添加覆盖物，使用 map.removeOverlay 方法移除覆盖物，注意此方法不适用于 InfoWindow。

标注

标注表示地图上的点。API 提供了默认图标样式，您也可以通过 Icon 类来指定自定义图标。Marker 的构造函数的参数为 Point 和 MarkerOptions（可选）。注意：当您使用自定义图标时，标注的地理坐标点将位于标注所用图标的中心位置，您可通过 Icon 的 offset 属性修改标定位置。

下面的示例向地图中心点添加了一个标注，并使用默认的标注样式。

```
var map = new BMap.Map("container");
var point = new BMap.Point(116.404, 39.915);
map.centerAndZoom(point, 15);
var marker = new BMap.Marker(point);        // 创建标注
map.addOverlay(marker);                     // 将标注添加到地图中
```

定义标注图标

通过 Icon 类可实现自定义标注的图标，下面示例通过参数 MarkerOptions 的 icon 属性进行设置，您也可以使用 marker.setIcon()方法。

```
var map = new BMap.Map("container");
var point = new BMap.Point(116.404, 39.915);
map.centerAndZoom(point, 15);
// 编写自定义函数，创建标注
function addMarker(point, index){
// 创建图标对象
var myIcon = new BMap.Icon("markers.png", new BMap.Size(23, 25), {
// 指定定位位置。
// 当标注显示在地图上时，其所指向的地理位置距离图标左上
   // 角各偏移10像素和25像素。您可以看到在本例中该位置即是
   // 图标中央下端的尖角位置。
   offset: new BMap.Size(10, 25),
   // 设置图片偏移。
   // 当您需要从一幅较大的图片中截取某部分作为标注图标时，您
   // 需要指定大图的偏移位置，此做法与 css sprites 技术类似。
   imageOffset: new BMap.Size(0, 0 - index * 25)   // 设置图片偏移
 });
 // 创建标注对象并添加到地图
 var marker = new BMap.Marker(point, {icon: myIcon});
 map.addOverlay(marker);
}
// 随机向地图添加10个标注
var bounds = map.getBounds();
var lngSpan = bounds.maxX - bounds.minX;
var latSpan = bounds.maxY - bounds.minY;
for (var i = 0; i < 10; i ++) {
 var point = new BMap.Point(bounds.minX + lngSpan * (Math.random() * 0.7 + 0.1
5),
                         bounds.minY + latSpan * (Math.random() * 0.7 + 0.15));

 addMarker(point, i);
}
```

```

```

监听标注事件

事件方法与 Map 事件机制相同。可参考事件部分。

```
marker.addEventListener("click", function(){
 alert("您点击了标注");
});
```

可托拽的标注

marker 的 enableDragging 和 disableDragging 方法可用来开启和关闭标注的拖拽功能。默认情况下标注不支持拖拽，您需要调用 marker.enableDragging()方法来开启拖拽功能。在标注开启拖拽功能后，您可以监听标注的 dragend 事件来捕获拖拽后标注的最新位置。

```
marker.enableDragging();
marker.addEventListener("dragend", function(e){
 alert("当前位置：" + e.point.lng + ", " + e.point.lat);
})
```

内存释放

在 API 1.0版本中，如果您需要在地图中反复添加大量的标注，这可能会占用较多的内存资源。如果您的标注在移除后不再使用，可调用 Overlay.dispose()方法来释放内存。注意在1.0版本中，调用此方法后标注将不能再次添加到地图上。自1.1版本开始，您不在需要使用此方法来释放内存资源，API 会自动帮助您完成此工作。

例如，您可以在标注被移除后调用此方法：

```
map.removeOverlay(marker);
marker.dispose(); // 1.1 版本不需要这样调用
```

信息窗口

信息窗口在地图上方的浮动显示 HTML 内容。信息窗口可直接在地图上的任意位置打开，也可以在标注对象上打开（此时信息窗口的坐标与标注的坐标一致）。 您可以使用 InfoWindow 来创建一个信息窗实例，注意同一时刻地图上只能有一个信息窗口处于打开状态。

```
var opts = {
 width : 250,     // 信息窗口宽度
```

```
 height: 100,     // 信息窗口高度
 title : "Hello"  // 信息窗口标题
}
var infoWindow = new BMap.InfoWindow("World", opts);  // 创建信息窗口对象
map.openInfoWindow(infoWindow, map.getCenter());      // 打开信息窗口
```

折线

Polyline 表示地图上的折线覆盖物。它包含一组点，并将这些点连接起来形成折线。

添加折线

折线在地图上绘制为一系列直线段。可以自定义这些线段的颜色、粗细和透明度。颜色可以是十六进制数字形式（比如：#ff0000）或者是颜色关键字（比如：red）。

Polyline 的绘制需要浏览器支持矢量绘制功能。在 Internet Explorer 中，地图使用 VML 绘制折线；在其他浏览器中使用 SVG 或者 Canvas

以下代码段会在两点之间创建6像素宽的蓝色折线：

```
var polyline = new BMap.Polyline([
   new BMap.Point(116.399, 39.910),
   new BMap.Point(116.405, 39.920)
 ],
 {strokeColor:"blue", strokeWeight:6, strokeOpacity:0.5}
);
map.addOverlay(polyline);
```

自定义覆盖物

API 自1.1版本起支持用户自定义覆盖物。

要创建自定义覆盖物，您需要做以下工作：

定义一个自定义覆盖物的构造函数，通过构造函数参数可以传递一些自由的变量。

设置自定义覆盖物对象的 prototype 属性为 Overlay 的实例，以便继承覆盖物基类。

实现 initialize 方法，当调用 map.addOverlay 方法时，API 会调用此方法。

实现 draw 方法。

定义构造函数并继承 Overlay

首先您需要定义自定义覆盖物的构造函数，在下面的示例中我们定义一个名为 SquareOverlay 的构造函数，

它包含中心点和边长两个参数，用来在地图上创建一个方形覆盖物。

```
// 定义自定义覆盖物的构造函数
function SquareOverlay(center, length, color){
 this._center = center;
 this._length = length;
 this._color = color;
}
// 继承 API 的 BMap.Overlay
SquareOverlay.prototype = new BMap.Overlay();
```

初始化自定义覆盖物

当调用 map.addOverlay 方法添加自定义覆盖物时，API 会调用该对象的 initialize 方法用来初始化覆盖物，在初始化过程中需要创建覆盖物所需要的 DOM 元素，并添加到地图相应的容器中。

地图提供了若干容器供覆盖物展示，通过 map.getPanes 方法可以得到这些容器元素，它们包括：

floatPane

markerMouseTarget

floatShadow

labelPane

markerPane

mapPane

这些对象代表了不同的覆盖物容器元素，它们之间存在着覆盖关系，最上一层为 floatPane，用于显示信息窗口内容，下面依次为标注点击区域层、信息窗口阴影层、文本标注层、标注层和矢量图形层。

我们自定义的方形覆盖物可以添加到任意图层上，这里我们选择添加到 markerPane 上，作为其一个子结点。

```
// 实现初始化方法
SquareOverlay.prototype.initialize = function(map){
// 保存 map 对象实例
 this._map = map;
 // 创建 div 元素，作为自定义覆盖物的容器
 var div = document.createElement("div");
 div.style.position = "absolute";
 // 可以根据参数设置元素外观
 div.style.width = this._length + "px";
 div.style.height = this._length + "px";
 div.style.background = this._color;
 // 将 div 添加到覆盖物容器中
 map.getPanes().markerPane.appendChild(div);
 // 保存 div 实例
```

```
 this._div = div;
 // 需要将 div 元素作为方法的返回值，当调用该覆盖物的 show、
 // hide 方法，或者对覆盖物进行移除时，API 都将操作此元素。
 return div;
}
```

绘制覆盖物

到目前为止，我们仅仅把覆盖物添加到了地图上，但是并没有将它放置在正确的位置上。您需要在 draw 方法中设置覆盖物的位置，每当地图状态发生变化（比如：位置移动、级别变化）时，API 都会调用覆盖物的 draw 方法，用于重新计算覆盖物的位置。通过 map.pointToOverlayPixel 方法可以将地理坐标转换到覆盖物的所需要的像素坐标。

```
// 实现绘制方法
SquareOverlay.prototype.draw = function(){
// 根据地理坐标转换为像素坐标，并设置给容器
 var position = this._map.pointToOverlayPixel(this._center);
 this._div.style.left = position.x - this._length / 2 + "px";
 this._div.style.top = position.y - this._length / 2 + "px";
}
```

移除覆盖物

当调用 map.removeOverlay 或者 map.clearOverlays 方法时，API 会自动将 initialize 方法返回的 DOM 元素进行移除。

显示和隐藏覆盖物

自定义覆盖物会自动继承 Overlay 的 show 和 hide 方法，方法会修改由 initialize 方法返回的 DOM 元素的 style.display 属性。如果自定义覆盖物元素较为复杂，您也可以自己实现 show 和 hide 方法。

```
// 实现显示方法
SquareOverlay.prototype.show = function(){
 if (this._div){
   this._div.style.display = "";
 }
}
// 实现隐藏方法
SquareOverlay.prototype.hide = function(){
 if (this._div){
   this._div.style.display = "none";
```

```
 }
}
```

自定义其他方法

通过构造函数的 prototype 属性，您可以添加任何自定义的方法，比如下面这个方法每调用一次就能改变覆盖物的显示状态：

```
// 添加自定义方法
SquareOverlay.prototype.toggle = function(){
 if (this._div){
   if (this._div.style.display == ""){
     this.hide();
   }
   else {
     this.show();
   }
 }
}
```

添加覆盖物

您现在已经完成了一个完整的自定义覆盖物的编写，可以添加到地图上了。

```
// 初始化地图
var map = new BMap.Map("container");
var point = new BMap.Point(116.404, 39.915);
map.centerAndZoom(point, 15);
// 添加自定义覆盖物
var mySquare = new SquareOverlay(map.getCenter(), 100, "red");
map.addOverlay(mySquare);
```

事件

地图事件概述

浏览器中的 JavaScript 是“事件驱动的”，这表示 JavaScript 通过生成事件来响应交互，并期望程序能够“监听”感兴趣的活动。例如，在浏览器中，用户的鼠标和键盘交互可以创建在 DOM 内传播的事件。对某些事件感兴趣的程序会为这些事件注册 JavaScript 事件监听器，并在接收这些事件时执行代码。

百度地图 API 拥有一个自己的事件模型，程序员可监听地图 API 对象的自定义事件，使用方法和 DOM 事件类似。但请注意，地图 API 事件是独立的，与标准 DOM 事件不同。

事件监听

百度地图 API 中的大部分对象都含有 addEventListener 方法，您可以通过该方法来监听对象事件。例如，BMap.Map 包含 click、dblclick 等事件。在特定环境下这些事件会被触发，同时监听函数会得到相应的事件参数 e，比如当用户点击地图时，e 参数会包含鼠标所对应的地理位置 point。

有关地图 API 对象的事件，请参考完整的 API 参考文档。

addEventListener 方法有两个参数：监听的事件名称和事件触发时调用的函数。在下面示例中，每当用户点击地图时，会弹出一个警告框。

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 11);
map.addEventListener("click", function(){
 alert("您点击了地图。");
});
```

通过监听事件还可以捕获事件触发后的状态。下面示例显示用户拖动地图后地图中心的经纬度信息。

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 11);
map.addEventListener("dragend", function(){
 var center = map.getCenter();
 alert("地图中心点变更为：" + center.lng + ", " + center.lat);
});
```

事件参数和 this

在标准的 DOM 事件模型中（DOM Level 2 Events），监听函数会得到一个事件对象 e，在 e 中可以获取有关该事件的信息。同时在监听函数中 this 会指向触发该事件的 DOM 元素。 百度地图 API 的事件模型与此类似，在事件监听函数中传递事件对象 e，每个 e 参数至少包含事件类型（type）和触发该事件的对象（target）。 API 还保证函数内的 this 指向触发（同时也是绑定）事件的 API 对象。

例如，通过参数 e 得到点击的经纬度坐标。

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 11);
map.addEventListener("click", function(e){
 alert(e.point.lng + ", " + e.point.lat);
});
```

或者通过 this 得到地图缩放后的级别。

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 11);
map.addEventListener("zoomend", function(){
 alert("地图缩放至：" + this.getZoom() + "级");
});
```

移除监听事件

当您不再希望监听事件时，可以将事件监听进行移除。每个 API 对象提供了 removeEventListener 用来移除事件监听函数。

下面示例中，用户第一次点击地图会触发事件监听函数，在函数内部对事件监听进行了移除，因此后续的点击操作则不会触发监听函数。

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 11);
function showInfo(e){
 alert(e.point.lng + ", " + e.point.lat);
 map.removeEventListener("click", showInfo);
}
map.addEventListener("click", showInfo);
```

地图图层

地图图层概念

地图可以包含一个或多个图层，每个图层在每个级别都是由若干张图块组成的，它们覆盖了地球的整个表面。例如您所看到包括街道、兴趣点、学校、公园等内容的地图展现就是一个图层，另外交通流量的展现也是通过图层来实现的。

目前百度地图提供的图层包括：

TrafficLayer：**交通流量图层**。

添加和移除图层

通过 map.addTileLayer 方法可向地图添加图层，例如下面代码将显示北京市的交通流量。

```
var map = new BMap.Map("container");          // 创建地图实例
var point = new BMap.Point(116.404, 39.915);  // 创建点坐标
map.centerAndZoom(point, 15);                 // 初始化地图，设置中心点坐标和地图级别
var traffic = new BMap.TrafficLayer();        // 创建交通流量图层实例
map.addTileLayer(traffic);                    // 将图层添加到地图上
```

若要从地图上移除图层，需要调用 map.removeTileLayer 方法。

```
map.removeTileLayer(traffic);                 // 将图层移除
```

自定义图层

地图坐标系

在使用自定义图层前，您需要了解百度地图的地图坐标系，百度地图坐标系涉及：

经纬度球面坐标系统

墨卡托平面坐标系统

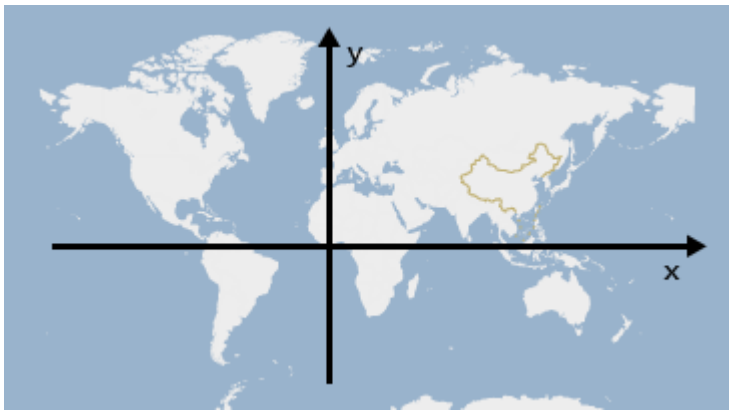
图块编号系统

经纬度是一种利用三维空间的球面来定义地球上的空间的球面坐标系，它能够标示地球上任何一个位置。通过伦敦格林尼治天文台原址的经线为0度经线，从0度经线向东、向西各分180度。赤道为0度纬线，赤道以北的纬线称为北纬、以南的称为南纬。在百度地图中，东经和北纬用正数标示，西经和南纬用负数标示。例如北京的位置大约是北纬39.9度，东经116.4度，那么用数值标示就是经度116.6，纬度39.9。在百度地

图中，习惯经度在前，纬度在后，例如：

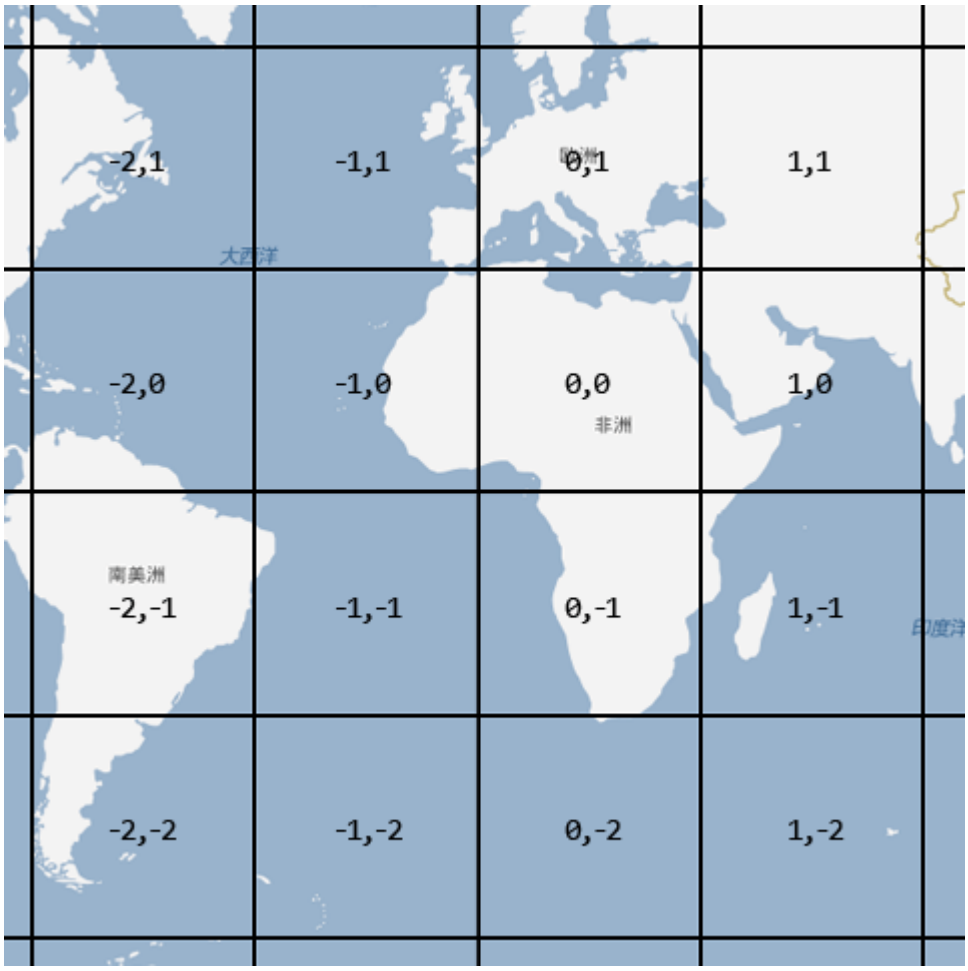
```
var point = new BMap.Point(116.404, 39.915);  // 创建点坐标，经度在前，纬度在后
```

由于百度地图是显示在平面上的，因此在地图内部系统中需要将球面坐标转换为平面坐标，这个转换过程称为投影。百度地图使用的是墨卡托投影。墨卡托平面坐标如下图所示，平面坐标与经纬度坐标系的原点是重合的。



百度地图在每一个级别将整个地图划分成若干个图块，通过编号系统将整个图块整合在一起以便显示完整的地图。当地图被拖动或者级别发生变化时，地图 API 将会根据平面坐标计算出当前视野内所需显示的图块的编号。

百度地图图块编号规则如下图所示：



从平面坐标原点开始的右上方向的图块编号为0,0，以此类推。在最低的缩放级别（级别 1）中，整个地球由 4 张图块组成。随着级别的增长，地图所使用的图块个数也随之增多。

定义取图规则

通过 TileLayer 类开发者可以实现自定义图层。其中，TileLayer 实例的 getTilesUrl 方法需要实现，用来告诉 API 取图规则。getTilesUrl 方法的参数包括 tileCoord 和 zoom，其中 tileCoord 为图块的编号信息，zoom 为图块的级别，每当地图需要显示特定级别的特定位置的图块时就会自动调用此方法，并提供这两个参数。使用者需要告知 API 特定编号和级别所对应的图块的地址，这样 API 就能正常显示自定义的图层了。

添加和移除自定义图层

以下代码在每个图块的所有缩放级别上显示一个简单的透明叠加层，使用浮动红色小水滴表示图块的轮廓。

```
var map = new BMap.Map("container");          // 创建地图实例
var point = new BMap.Point(116.404, 39.915);  // 创建点坐标
map.centerAndZoom(point,15);                 // 初始化地图，设置中心点坐标和地图级别
var tilelayer = new BMap.TileLayer();         // 创建地图层实例
tilelayer.getTilesUrl=function(){             // 设置图块路径
```

```
 return "layer.gif";
};
map.addTileLayer(tilelayer);                    // 将图层添加到地图上
```

工具

地图工具概述

百度地图提供了交互功能更为复杂的“工具”，它包括：

PushpinTool：标注工具。通过此工具用户可在地图任意区域添加标注。

DistanceTool：测距工具。通过此工具用户可测量地图上任意位置之间的距离。

DragAndZoomTool：区域缩放工具。此工具将根据用户拖拽绘制的矩形区域大小对地图进行放大或缩小操作。

工具类在初始化时需要提供地图实例参数，以便使工具在该地图上生效。您可以在地图上添加多个工具，但同一时刻只能有一个工具处于开启状态。标注工具和测距工具在完成一次操作后将自动退出开启状态，而区域缩放工具可以自行配置是否自动关闭。

向地图添加工具

在地图正确初始化后，您可以创建工具实例。下面示例展示了如何向地图添加一个标注工具。

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 15);
var myPushpin = new BMap.PushpinTool(map);          // 创建标注工具实例
myPushpin.addEventListener("markend", function(e){  // 监听事件,提示标注点坐标信息

 alert("您标注的位置：" +
      e.marker.getPoint().lng + ", " +
      e.marker.getPoint().lat);
});
myPushpin.open();                                   // 开启标注工具
```

通过按钮控制工具的开启和关闭

工具类没有提供控制其开启和关闭的 UI 元素。您可以根据需要自己创建这些元素，把它们放置在地图区域内或者区域外均可。调用工具类的 open 和 close 可控制工具的开启和关闭。

首先初始化地图并创建一个测距工具实例：

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 15);
var myDis = new BMap.DistanceTool(map);
```

接着我们创建两个按钮元素并为其添加点击事件。

```
<input type="button" value="开启" onclick="myDis.open()" />
<input type="button" value="关闭" onclick="myDis.close()" />
```

拉框放大工具

一些工具类提供了可修改的配置参数，您可参考 API 文档来修改它们以便符合您的要求。

本示例为区域缩放工具添加提示文字。

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 15);
var myDrag = new BMap.DragAndZoomTool(map, {
 followText : "拖拽鼠标进行操作"
});
```

服务

地图服务概述

地图服务是指那些提供数据信息的接口，比如本地搜索、路线规划等等。百度地图 API 提供的服务有：

LocalSearch：本地搜索，提供某一特定地区的位置搜索服务，比如在北京市搜索“公园”。

TransitRoute：公交导航，提供某一特定地区的公交出行方案的搜索服务。

DrivingRoute：驾车导航，提供驾车出行方案的搜索服务。

WalkingRoute：步行导航，提供步行出行方案的搜索服务。

Geocoder：地址解析，提供将地址信息转换为坐标点信息的服务。

LocalCity：本地城市，提供自动判断您所在城市的服务。

TrafficControl：实时路况控件，提供实时和历史路况信息服务。

搜索类的服务接口需要指定一个搜索范围，否则接口将不能工作。

本地搜索

BMap.LocalSearch 提供本地搜索服务，在使用本地搜索时需要为其设置一个检索区域，检索区域可以是 BMap.Map 对象、BMap.Point 对象或者是省市名称（比如："北京市"）的字符串。BMap.LocalSearch 构造函数的第二个参数是可选的，您可以在其中指定结果的呈现。BMap.RenderOptions 类提供了若干控制呈现的属性，其中 map 指定了结果所展现的地图实例，panel 指定了结果列表的容器元素。

下面这个示例展示了在北京市检索天安门。搜索区域设置为地图实例，并告知结果需要展现在地图实例上。

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 11);
var local = new BMap.LocalSearch(map, {
 renderOptions:{map: map}
});
local.search("天安门");
```

另外，BMap.LocalSearch 还提包含 searchNearby 和 searchInBounds 方法，为您提供周边搜索和范围搜索服务。

配置搜索

BMap.LocalSearch 提供了若干配置方法，通过它们可以自定义搜索服务的行为以满足您的需求。

在下面的示例中，我们调整每页显示8个结果，并且根据结果点位置自动调整地图视野，不显示第一条结果的信息窗口：

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 11);
var local = new BMap.LocalSearch("北京市", {
 renderOptions: {
   map: map,
   autoViewport: true,
   selectFirstResult: false
 },
  pageCapacity: 8
});
```

```
local.search("中关村");
```

结果面板

通过设置 BMap.LocalSearchOptions.renderOptions.panel 属性，可以为本地搜索对象提供一个结果列表容器，搜索结果会自动添加到容器元素中。请看下面示例：

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 11);
var local = new BMap.LocalSearch(map, {
 renderOptions: {map: map, panel: "results"}
});
local.search("天安门");
```

数据接口

除了搜索结果会自动添加到地图和列表外，您还可以通过数据接口获得详细的数据信息，结合地图 API 您可以自行向地图添加标注和信息窗口。BMap.LocalSearch 和 BMap.LocalSearchOptions 类提供了若干设置回调函数的接口，通过它们可得到搜索结果的数据信息。 例如，通过 onSearchComplete 回调函数参数可以获得 BMap.LocalResult 对象实例，它包含了每一次搜索结果的数据信息。 当回调函数被执行时，您可以使用 BMap.LocalSearch.getStatus()方法来确认搜索是否成功或者得到错误的详细信息。

在下面这个示例中，通过 onSearchComplete 回调函数得到第一页每条结果的标题和地址信息，并输出到页面上：

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 11);
var options = {
 onSearchComplete: function(results){
   if (local.getStatus() == BMAP_STATUS_SUCCESS){
     // 判断状态是否正确
     var s = [];
     for (var i = 0; i < results.getCurrentNumPois(); i ++){
       s.push(results.getPoi(i).title + ", " + results.getPoi(i).address);
     }
     document.getElementById("log").innerHTML = s.join("
");
   }
 }
};
var local = new BMap.LocalSearch(map, options);
local.search("公园");
```

周边搜索

通过周边搜索服务，您可以在某个地点附近进行搜索，也可以在某一个特定结果点周围进行搜索。

下面示例展示如何在前门附近搜索小吃：

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 11);
var local = new BMap.LocalSearch(map, {
 renderOptions:{map: map, autoViewport: true}
});
local.searchNearby("小吃", "前门");
```

范围搜索

范围搜索将根据您提供的视野范围提供搜索结果。注意：当搜索范围过大时可能会出现无结果的情况。

下面示例展示在当前地图视野范围内搜索银行：

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 14);
var local = new BMap.LocalSearch(map, {
 renderOptions:{map: map}
});
local.searchInBounds("银行", map.getBounds());
```

公交导航

BMap.TransitRoute 类提供公交导航搜索服务。和本地搜索类似，在搜索之前需要指定搜索区域，注意公交导航的区域范围只能是市，而不能是省。如果搜索区域为 BMap.Map 对象，路线结果会自动添加到地图上。如果您提供了结果容器，相应的路线描述也会展示在页面上。

下面示例展示了如何使用公交导航服务：

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 14);
var transit = new BMap.TransitRoute(map, {
 renderOptions: {map: map}
});
transit.search("王府井", "西单");
```

结果面板

您可以提供用于展示文字结果的容器元素，方案结果会自动在页面中展现：

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 14);
var transit = new BMap.TransitRoute(map, {
 renderOptions: {map: map, panel: "results"}
});
transit.search("王府井", "西单");
```

数据接口

您可通过数据接口获取详细的公交方案信息。公交导航搜索结果用 BMap.TransitRouteResult 来表示，其中包含了若干公交出行方案（BMap.TransitRoutePlan）。每条出行方案由步行线路和公交线路组成。 在起点到上车点之间、下车点到终点之间以及每个换乘站之间都会存在步行线路，如果上述的某两点位置重合，那么其间的步行路线长度为0。

在下面示例中，通过数据接口将第一条方案的路线添加到地图上，并将所有方案的描述信息输出到页面上

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 12);
var transit = new BMap.TransitRoute("北京市");
transit.setSearchCompleteCallback(function(results){
 if (transit.getStatus() == BMAP_STATUS_SUCCESS){
   var firstPlan = results.getPlan(0);
   // 绘制步行线路
   for (var i = 0; i < firstPlan.getNumRoutes(); i ++){
     var walk = firstPlan.getRoute(i);
     if (walk.getDistance(false) > 0){
       // 步行线路有可能为0
       map.addOverlay(new BMap.Polyline(walk.getPoints(), {lineColor: "green
"}));
     }
   }
   // 绘制公交线路
   for (i = 0; i < firstPlan.getNumLines(); i ++){
     var line = firstPlan.getLine(i);
     map.addOverlay(new BMap.Polyline(line.getPoints()));
   }
   // 输出方案信息
   var s = [];
   for (i = 0; i < results.getNumPlans(); i ++){
     s.push((i + 1) + ". " + results.getPlan(i).getDescription());
```

```
   }
   document.getElementById("log").innerHTML = s.join("
");
 }
})
transit.search("中关村", "国贸桥");
```

驾车导航

BMap.DrivingRoute 提供驾车导航服务。与公交导航不同的是，驾车导航的搜索范围可以设置为省。

下面示例展示了如何使用驾车导航接口：

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 14);
var driving = new BMap.DrivingRoute(map, {
 renderOptions: {
   map: map,
   autoViewport: true
 }
});
driving.search("中关村", "天安门");
```

结果面板

下面示例中，我们提供了结果面板参数，方案描述会自动展示到页面上。

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 14);
var driving = new BMap.DrivingRoute(map, {
 renderOptions: {
   map   : map,
   panel : "results",
   autoViewport: true
 }
});
driving.search("中关村", "天安门");
```

数据接口

驾车导航服务也提供了丰富的数据接口，通过 onSearchComplete 回调函数可以得到

BMap.DrivingRouteResult 对象，它包含了驾车导航结果数据信息。 结果会包含若干驾车方案（目前仅提供一条方案），每条方案中包含了若干驾车线路（如果导航方案只包含一个目的地，那么驾车线路的个数就为1，如果方案包含若干个目的地，则驾车线路的个数会大于1。目前 API 尚不支持多个目的地的驾车导航）。每条驾车线路又会包含一系列的关键步骤（BMap.Step），关键步骤描述了具体驾车行驶方案，可通过BMap.Step.getDescription()方法获得。

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 14);
var options = {
 onSearchComplete: function(results){
   if (driving.getStatus() == BMAP_STATUS_SUCCESS){
     // 获取第一条方案
     var plan = results.getPlan(0);
     // 获取方案的驾车线路
     var route = plan.getRoute(0);
     // 获取每个关键步骤，并输出到页面
     var s = [];
     for (var i = 0; i < route.getNumSteps(); i ++){
       var step = route.getStep(i);
       s.push((i + 1) + ". " + step.getDescription());
     }
     document.getElementById("log").innerHTML = s.join("
");
   }
 }
};
var driving = new BMap.DrivingRoute(map, options);
driving.search("中关村", "天安门");
```

步行导航接口在使用上与驾车导航一致，具体请参考 API 文档。

地理编码

地理编码能够将地址信息转换为地理坐标点信息。

根据地址描述获得坐标

百度地图 API 提供 Geocoder 类进行地址解析，您可以通过 Geocoder.getPoint()方法来将一段地址描述转换为一个坐标。

在下面的示例中，我们将获得地址“北京市海淀区上地10街10号”的地理坐标位置，并在这个位置上添加一个标注。注意在调用 Geocoder.getPoint()方法时您需要提供地址解析所在的城市（本例为“北京市”）。

```
var map = new BMap.Map("container");
```

```
map.centerAndZoom(new BMap.Point(116.404, 39.915), 11);
// 创建地址解析器实例
var myGeo = new BMap.Geocoder();
// 将地址解析结果显示在地图上，并调整地图视野
myGeo.getPoint("北京市海淀区上地10街10号", function(point){
 if (point) {
   map.centerAndZoom(point, 16);
   map.addOverlay(new BMap.Marker(point));
 }
}, "北京市");
```

反向地理编码

反向地理编码的过程正好相反，它根据一个坐标点得到一个地址的描述。您可以通过Geocoder.getLocation()方法获得地址描述。当解析工作完成后，您提供的回调函数将会被触发。如果解析成功，则回调函数的参数为 GeocoderResult 对象，否则为 null。

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 11);
// 创建地理编码实例
var myGeo = new BMap.Geocoder();
// 根据坐标得到地址描述
myGeo.getLocation(new BMap.Point(116.364, 39.993), function(result){
 if (result){
   alert(result.address);
 }
});
```