# HTML5 第五天笔记

## 百度地图

### 1. 如何使用

#### 1）HTML 页面

* 引入百度地图提供的JS文件

<script src="http://api.map.baidu.com/api?v=2.0&ak=HbUVYMUg6PwbOnXkztdgSQlQ"></script>

* 定义容器元素<div></div>

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>百度地图如何使用</title>

<style type="text/css">

body, html,#allmap {

width: 100%;

height: 100%;

overflow: hidden;

margin:0;

font-family:"微软雅黑";

}

</style>

<!--

引入百度地图提供的JS文件

\* 是在线提供的JS文件 - 必须可以联网

http://api.map.baidu.com/api?v=2.0&ak=您的密钥

\* v=2.0 - 使用的百度地图的版本

\* ak=您的密钥 - 传递当前应用的秘钥

\* 只有秘钥正确,正常地使用百度地图的功能

-->

<script src="http://api.map.baidu.com/api?v=2.0&ak=HbUVYMUg6PwbOnXkztdgSQlQ"></script>

</head>

<body>

<!-- 定义一个容器元素 - 用于显示百度地图 -->

<div id="allmap"></div>

</body>

</html>

#### 2）JavaScript 逻辑

* 创建百度地图对象

var map = new BMap.Map(容器元素id);

* 初始化百度地图

map.centerAndZoom(中心点坐标,显示级别)

如下述示例代码：

/\*

1. 通过Map()构造函数,创建百度地图对象

var map = new BMap.Map(容器元素id属性值);

\*/

var map = new BMap.Map("allmap");

/\*

2. (必要)初始化百度地图 - 设置中心点和显示级别

map.centerAndZoom(center,zoom)

\* center - 设置百度地图当前的中心点坐标

\* Point类型,zoom参数必须设置

\* String类型,zoom参数可有可无

\* 如果不设置zoom,自动匹配最佳显示级别

\* zoom - 设置百度地图当前的显示级别

\* 一般情况下,值范围为 3-19

\* 高清地图(移动端),值范围为 3-18

\*/

map.centerAndZoom("北京",12);

### 2. 核心类

#### Map类

**构造器**

| **构造函数** | **参数** | **说明** |
| --- | --- | --- |
| Map(container) | container：页面容器元素id | 创建地图实例对象 |

**配置方法**

| **方法名称** | **参数** | **说明** |
| --- | --- | --- |
| enableScrollWheelZoom() |  | 开启鼠标滚轮放大或缩小地图的显示级别 |
| disableScrollWheelZoom() |  | 禁用鼠标滚轮功能 |
| enableDragging() |  | 启用鼠标拖拽功能 |
| disableDragging() |  | 禁用鼠标拖拽功能 |

**地图状态方法**

| **方法名称** | **参数** | **说明** |
| --- | --- | --- |
| getCenter() |  | 返回地图当前中心点(Point类型) |
| getZoom() |  | 返回地图当前缩放级别(Number类型) |

**修改地图状态方法**

| **方法名称** | **参数** | **说明** |
| --- | --- | --- |
| centerAndZoom(center,zoom) | center：设置地图的中心点坐标；zoom：设置地图的显示级别 | 初始化地图中心点和级别 |
| setCenter(center) | center：设置地图的中心点坐标 | 设置地图中心点(Point |
| setCurrentCity(city) | city：设置地图城市 | 设置地图城市 |
| setZoom(zoom) | zoom：设置地图的显示级别 | 设置显示级别 |

**控件方法**

| **方法名称** | **参数** | **说明** |
| --- | --- | --- |
| addControl(control) |  | 将控件添加到地图 |
| removeControl(control) |  | 从地图中移除控件 |

**覆盖物方法**

| **方法名称** | **参数** | **说明** |
| --- | --- | --- |
| addOverlay(overlay) |  | 将覆盖物添加到地图中 |
| removeOverlay(overlay) |  | 从地图中移除覆盖物 |
| openInfoWindow(InfoWindow, Point) | InfoWindow：设置信息窗口；Point：在指定点打开 | 设置信息窗口 |

### 3. 控件类

#### ScaleControl类

表示比例尺控件。

**构造器**

| **构造函数** | **参数** | **说明** |
| --- | --- | --- |
| ScaleControl(ScaleControlOptions) | - | 创建地图比例尺对象 |

#### ScaleControlOptions类

表示ScaleControl构造函数的可选参数。

**选项**

* anchor：设置当前控件显示的位置。
  + BMAP\_ANCHOR\_TOP\_LEFT：左上角
  + BMAP\_ANCHOR\_BOTTOM\_LEFT：左下角
  + BMAP\_ANCHOR\_TOP\_RIGHT：右上角
  + BMAP\_ANCHOR\_BOTTOM\_RIGHT：右下角

#### NavigationControl类

表示地图的平移缩放控件。

**构造器**

| **构造函数** | **参数** | **说明** |
| --- | --- | --- |
| NavigationControl(NavigationControlOptions) | - | 创建地图平移缩放对象 |

#### MapTypeControl类

负责切换地图类型的控件。

**构造器**

| **构造函数** | **参数** | **说明** |
| --- | --- | --- |
| MapTypeControl(MapTypeControlOptions) | - | 创建地图类型对象 |

#### OverviewMapControl类

表示缩略地图控件。

**构造器**

| **构造函数** | **参数** | **说明** |
| --- | --- | --- |
| OverviewMapControl(OverviewMapControlOptions) | - | 创建缩略地图对象 |

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>百度地图的控件类</title>

<style type="text/css">

body, html,#allmap {

width: 100%;

height: 100%;

overflow: hidden;

margin:0;

font-family:"微软雅黑";

}

</style>

<script src="http://api.map.baidu.com/api?v=2.0&ak=HbUVYMUg6PwbOnXkztdgSQlQ"></script>

</head>

<body>

<div id="allmap"></div>

<script>

// 1. 创建地图对象

var map = new BMap.Map("allmap");

// 2. 初始化地图

map.centerAndZoom("北京",13);

map.enableScrollWheelZoom();

/\*

3. 添加比例尺控件

\* 创建比例尺控件对象

\* 将比例尺控件添加到地图中

\*/

var scale = new BMap.ScaleControl({

anchor: BMAP\_ANCHOR\_TOP\_RIGHT

});

map.addControl(scale);

/\*

4. 添加平移缩放控件

\* 创建平移缩放控件对象

\* 将平移缩放控件添加到地图中

\*/

var nav = new BMap.NavigationControl({

anchor: BMAP\_ANCHOR\_TOP\_RIGHT

});

map.addControl(nav);

// 5. 添加地图类型切换控件

var type = new BMap.MapTypeControl({

anchor: BMAP\_ANCHOR\_TOP\_LEFT

});

map.addControl(type);

// 6. 添加缩略地图控件

var overview = new BMap.OverviewMapControl({

isOpen : true//表示是否自动打开

});

map.addControl(overview);

</script>

</body>

</html>

### 4. 覆盖物类

#### Marker类

表示地图上一个图像标注。

**构造器**

| **构造函数** | **参数** | **说明** |
| --- | --- | --- |
| Marker(point) | - | 创建标注对象 |

#### InfoWindow类

表示地图上包含信息的窗口。

**构造器**

| **构造函数** | **参数** | **说明** |
| --- | --- | --- |
| InfoWindow(content) | content：设置当前点的地址信息内容 | 创建信息的窗口对象 |

**方法**

| **方法名称** | **参数** | **说明** |
| --- | --- | --- |
| setWidth(width) |  | 设置信息窗口的宽度 |
| setHeight(height) |  | 设置信息窗口的高度 |
| setTitle(title) |  | 设置信息窗口的标题 |

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>为百度地图进行标注</title>

<style type="text/css">

body, html,#allmap {

width: 100%;

height: 100%;

overflow: hidden;

margin:0;

font-family:"微软雅黑";

}

</style>

<script src="http://api.map.baidu.com/api?v=2.0&ak=HbUVYMUg6PwbOnXkztdgSQlQ"></script>

</head>

<body>

<div id="allmap"></div>

<script>

var map = new BMap.Map("allmap");

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

map.centerAndZoom(point, 15);

/\*

如何进行标注

\* 创建标注对象

\* 将标注添加到地图中

\*/

var marker = new BMap.Marker(point);

map.addOverlay(marker);

marker.setAnimation(BMAP\_ANIMATION\_BOUNCE);

</script>

</body>

</html>

### 5. 服务类

#### Geocoder类

用于获取用户的地址解析。

**构造器**

| **构造函数** | **参数** | **说明** |
| --- | --- | --- |
| Geocoder() | content：设置当前点的地址信息内容 | 创建地址解析对象 |

**方法**

| **方法名称** | **参数** | **说明** |
| --- | --- | --- |
| getPoint(address,callback,city) | address：要解析的地址内容；callback：回调函数（解析address地址成功的话,回调函数具有一个形参）；city：当前地址所在的城市名称 | 对指定的地址进行解析 |

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>百度地图案例</title>

<style type="text/css">

body, html,#allmap {

width: 100%;

height: 100%;

overflow: hidden;

margin:0;

font-family:"微软雅黑";

}

</style>

<script src="http://api.map.baidu.com/api?v=2.0&ak=HbUVYMUg6PwbOnXkztdgSQlQ"></script>

</head>

<body>

<div id="allmap"></div>

<script>

// 1. 创建地图对象

var map = new BMap.Map("allmap");

// 2. 初始化地图

//map.centerAndZoom("北京",13);

// 3. 启用鼠标的滚轮功能

map.enableScrollWheelZoom();

/\*

4. 根据给定的地址获取Point的经度和纬度

var geocoder = new BMap.Geocoder();

geocoder.getPoint(address,function(point){},city);

\*/

var geocoder = new BMap.Geocoder();

geocoder.getPoint("北京市海淀区万寿路西街2号",function(point){

// 5. 根据Point进行标注

var marker = new BMap.Marker(point);

map.addOverlay(marker);

// 6. 重新设置中心点位置

map.centerAndZoom(point,16);

// 7. 设置信息窗口

var opts = {

width : 200,

height: 100,

title : "文博大厦"

}

var info = new BMap.InfoWindow("地址:北京市海淀区万寿路西街2号",opts);

marker.addEventListener("click",function(){

map.openInfoWindow(info,point);

});

},"北京");

</script>

</body>

</html>

## 拖放 API

允许用户在一个元素上点击并按住鼠标按钮，拖动它到别的位置，然后松开鼠标按钮将元素放到那儿。

在拖动操作过程中，被拖动元素会以半透明形式展现，并跟随鼠标指针移动。放置元素的位置可能会在不同的应用内。

### 1. 源元素事件

**所谓源元素**就是被拖动的元素。

| **事件名称** | **作用** | **触发次数** |
| --- | --- | --- |
| dragstart事件 | 开始拖动源元素时被触发 | 只被触发一次 |
| drag事件 | 拖动的过程中，实时被触发 | 被触发多次 |
| dragend事件 | 结束拖动源元素时被触发 | 只被触发一次 |

<div id="d1">

<img id="img" src="Penguins.jpg" width="256">

</div>

<script>

// 获取HTML页面中的源元素<img>

var img = document.getElementById("img");

// 为源元素绑定拖动事件

img.addEventListener("dragstart",myDragstart);

img.addEventListener("drag",myDrag);

img.addEventListener("dragend",myDragend);

// 定义事件的处理函数

function myDragstart(event){

console.log("开始拖动啦...");

}

function myDrag(event){

console.log("拖动过程中...");

}

function myDragend(event){

console.log("结束拖动啦...");

}

</script>

### 2. 目标元素事件

**所谓目标元素**就是投放到的元素。

| **事件名称** | **作用** | **触发次数** |
| --- | --- | --- |
| dragenter事件 | 当源元素到达目标元素时被触发 | 只被触发一次 |
| dragover事件 | 当源元素到达目标元素时被触发 | 被触发多次。阻止默认行为，触发drop事件 |
| drop事件 | 当源元素投放在目标元素时被触发 | 默认该事件不会被触发 |
| dragleave事件 | 当源元素离开目标元素时被触发 | 只被触发一次 |

<div id="d1">

<img id="img" src="Penguins.jpg" width="256">

</div>

<div id="d2"></div>

<script>

// 获取目标元素

var d2 = document.getElementById("d2");

// 为目标元素绑定事件

d2.addEventListener("dragenter",myDragenter);

d2.addEventListener("dragover",myDragover);

d2.addEventListener("drop",myDrop);

d2.addEventListener("dragleave",myDragleave);

// 定义事件的处理函数

function myDragenter(event){

//event.preventDefault();

console.log("大爷,你来啦...");

}

function myDragover(event){

event.preventDefault();

console.log("大爷,又来啦...");

}

function myDrop(event){

//event.preventDefault();

console.log("大爷,别走啦...");

}

function myDragleave(event){

//event.preventDefault();

console.log("大爷,要走啦...");

}

</script>

### 3. dataTransfer对象

该对象已被集成在 event 对象中，起到剪切板的功能。

| **方法名称** | **作用** | **场景** |
| --- | --- | --- |
| setData(type,data) | 存储数据 | 在源元素事件中 |
| getData(type) | 取出数据 | 在目标元素事件中 |
| clearData() | 清除剪切板中所有数据 |  |

参数：

* type：指定当前存储数据的类型(标识)。
* data：需要中转的数据内容。

<div id="d1">

<img id="img" src="Penguins.jpg" width="256">

</div>

<div id="d2"></div>

<script>

// 1. 获取源元素和目标元素

var img = document.getElementById("img");

var d2 = document.getElementById("d2");

/\*

2. 为源元素和目标元素绑定事件

\* 绑定源元素事件的目的:

\* 通过dataTransfer对象获取源元素的关键数据

\* 只需绑定dragstart事件即可

\* 绑定目标元素事件的目的:

\* 通过dataTransfer对象使用源元素的关键数据

\* 需要绑定dragover和drop事件

\*/

img.addEventListener("dragstart",function(event){

// 通过dataTransfer对象,存储源元素的关键数据

event.dataTransfer.setData("text",img.src);

});

d2.addEventListener("dragover",function(event){

event.preventDefault();

});

d2.addEventListener("drop",function(event){

// 通过dataTransfer对象,使用源元素的关键数据

var src = event.dataTransfer.getData("text");

//d2.innerHTML = "<img src='"+src+"' width='256'>";

var newImg = document.createElement("img");

newImg.src = src;

newImg.width = "256";

newImg.id = "newImg";

d2.appendChild(newImg);

img.parentNode.removeChild(img);

});

</script>

**原创声明：**如发布或转载请注明以下作者信息，谢谢！

本文作者 - 金云龙

联系方式 - QQ：2080954535；微信：imlonge；微信公众号：longestory

个人网站：<http://www.longestory.com>