Table of Contents

WMap API 参考文档

WMap API 用来在浏览器中显示可交互的电子地图。

支持的浏览器包括：IE8-11、Chrome、Firefox、Android browser 2.2+, 3.1+, 4+、Safari for iOS 7+

Map

继承Evented

基本用法

```
js //在id为 mapDivId 的div中初始化地图，并设置中心点和初始化级别 var map = W.map('mapDivId', { center: [30.29, 111.34], zoom: 12 });
```

构造函数

方法	描述
W.map(< String > id, < Map options> options?)	通过一个给定的div标签的id和地图选项来创建一个地图对象。
W.map(< HTMLElement > el), < Map options> options?)	通过一个html element对象和地图选项来创建地图对象。

地图选项

选项名	类型	默认值	描述
attributionControl	Boolean	true	指定是否将属性控件添加到地图上。
zoomControl	Boolean	true	指定是否将级别缩放控件添加到地图上。
closePopupOnClick	Boolean	true	当设置为true的时候,鼠标点击地图（不是气泡的关闭按钮）将关闭已打开的气泡
minZoom	Number	W.config.getMinZoom()	地图最小级别
maxZoom	Number	W.config.getMaxZoom()	地图最大级别
maxBounds	LatLngBounds	W.config.getMaxBounds()	地图范围
doubleClickZoom	Boolean	true	当双击地图时,是否放大地图

事件

图层相关事件

事件名称	数据	描述
baselayerchange	LayersControlEvent	当底图改变时触发.
overlayadd	LayersControlEvent	当在图层控制控件中选择显示一个图层时触发该事件。
overlayremove	LayersControlEvent	当在图层控制控件中选择隐藏一个图层时触发该事件。

layeradd	LayerEvent	当向地图上添加一个图层时触发
layerremove	LayerEvent	当从地图上删除一个图层时触发

地图状态改变事件

事件名称	数据	描述
zoomlevelschange	Event	当地图级别改变时触发.
resize	ResizeEvent	当地图大小发生改变的时候触发.
unload	Event	当地图销毁时触发.
viewreset	Event	当地图上的内容需要重绘时触发，一般是在地图级别发生改变，或地图加载时触发.
load	Event	当地图第一次加载时触发.
zoomstart	Event	当地图级别发生改变前触发.
movestart	Event	当地图移动开始时触发.
zoom	Event	当地图级别改变时触发.
move	Event	当地图移动时触发.
zoomend	Event	当地图级别发生改变时触发.
moveend	Event	当地图移动后触发.

气泡相关事件

事件名称	回调数据	描述
popupopen	PopupEvent	当气泡打开时触发
popupclose	PopupEvent	当气泡关闭时触发
autopanstart	Event	当地图自动移动时触发，一般都是在弹出气泡的时候触发

交互相关事件

事件名称	回调数据	描述
click	MouseEvent	当点击地图时触发.
dblclick	MouseEvent	当双击地图时触发.
mousedown	MouseEvent	当鼠标落下时触发.
mouseup	MouseEvent	当鼠标弹起时触发.
mouseover	MouseEvent	当鼠标移入时触发.
mouseout	MouseEvent	当鼠标移出触发.
mousemove	MouseEvent	当鼠标移动时触发
contextmenu	MouseEvent	当点击右键时触发.
keypress	KeyboardEvent	当点击键盘上的按钮时触发.

方法

图层和控件相关方法

方法名	返回值	描述
addControl(< Control > control)	this	向地图上添加控件
removeControl(< Control > control)	this	从地图上删除控件
addLayer(< Layer >layer)	this	将图层添加到地图上
removeLayer< Layer >layer	this	从地图上删除图层.
hasLayer< Layer >layer	Boolean	判断图层是否已经叠加到地图上
openPopup(< Popup >popup)	this	在地图上打开一个气泡.
closePopup()	this	关闭地图上的气泡

修改地图状态相关方法

方法名	返回值	描述
setView(<LatLng> center,< Number > zoom)	this	设置地图视角（中心点和级别）
setZoom(< Number >zoom)	this	设置地图级别.
zoomIn()	this	放大地图
zoomOut()	this	缩小地图
fitBounds(< LatLngBounds > bounds)	this	固定地图范围.
fitWorld(<fitBounds options`>options?)	this	固定地图范围.
panTo(<LatLng> latlng)	this	平移地图至指定的坐标.
panBy(<Point> offset)	this	平移地图到指定的像素点.
flyTo(<LatLng>latlng,<Number>zoom?)	this	平滑的跳转到指定的位置和级别.
flyToBounds(<LatLngBounds> bounds)	this	平滑跳转到指定的范围.
setMaxBounds(<Bounds >bounds)	this	设置地图最大范围
setMinZoom(< Number > zoom)	this	设置地图最小级别.
setMaxZoom(< Number > zoom)	this	设置地图最大级别.
stop()	this	停止地图正在进行的行为panTo、FlyTo等.

获得地图状态相关方法

方法名	返回值	描述
getCenter()	LatLng	获得地图中心点
getZoom()	Number	获得地图级别
getBounds()	LatLngBounds	获得地图当前范围

getMinZoom()	Number	获得地图最小级别.
getMaxZoom()	Number	获得地图最大级别.
getBoundsZoom(< LatLngBounds >bounds, <Boolean>inside?)	Number	获得填充指定范围的地图最大级别.
getSize()	Point	获得地图的像素大小.
getPixelBounds()	Bounds	获得地图当前的像素大小范围.
getPixelOrigin()	Point	获得地图像素范围大小从原点开始计算.
getPixelWorldBounds(< Number >zoom?)	Bounds	获得地图的像素范围

绘制相关方法

方法名	返回值	描述
drawPoint(< Function > callback)	this	绘制点 该回调函数有一个参数：geom:该点的坐标
drawCircle(< Function > callback)	this	绘制圆 该回调函数有两个参数：geom：圆心的坐标,radius：圆的半径
drawRectangle(< Function > callback)	this	绘制矩形 该回调函数有一个参数：geom:该矩形的坐标
drawLineString(< Function > callback)	this	绘制折线 该回调函数有一个参数：geom:该折线的坐标
drawPolygon(< Function > callback)	this	绘制多边形 该回调函数有一个参数：geom:该多边形的坐标
stopDraw()	this	停止绘制
measureArea(< Function > callback)	this	绘制一个面并且测量它的面积,该回调函数有两个参数：area:该面的面积 没有可读性（单位：平方米）, readableArea 该面的面积 有可读性（单位：平方千米）
measureLength(< Function > callback)	this	绘制一个折线并且测量它的长度,该回调函数有两个参数：length：该折线的长度 没有可读性（单位：米）,readableLength 该折线的长度 有可读性(单位：千米)

GeoJSON相关相关方法

方法名	返回值	描述
geometryToLayer(< Object > geojson,< Options> options?)	Layer	将GeoJSON要素或几何对象转换为Layer
geometryToCircle(< Object > geojson,< Number> redius,< Options> options?)	Circle	将GeoJSON要素转换为一个Circle Layer
geometryToMarker(< Object > geojson,< Options> options?)	Marker	将GeoJSON要素转换为Marker

geometryToCircleMarker(< Object > geojson,< Options> options?)	CircleMarker	将GeoJSON要素转换为CircleMarker
asFeature(< Object > geom)	Feature	将GeoJSON几何对象转换为GeoJSON 要素对象
asGeometry(< Object > feature)	Geometry	将GeoJSON要素对象转换为GeoJSON 几何对象

其他

方法名	返回值	描述
invalidateSize()	this	如果地图容器的大小发生变化,调用这个方法可以动态的改变地图的大小

Layer

继承Evented。

所有图层类的父类。

方法

方法	返回值	描述
addTo(< Map > type)	this	将图层添加到地图中
remove()	this	从地图中移除图层
bindPopup (< String / HTMLElement / Function /Popup > content, options?)	this	绑定一个气泡
unbindPopup()	this	取消气泡绑定
openPopup()	this	打开气泡
closePopup()	this	关闭气泡
isPopupOpen()	this	气泡是否打开
setPopupContent(< String / HTMLElement / Popup > content)	this	修改气泡内容

MSPTileLayer

继承自：Layer

MSP栅格瓦片图层 。

示例

```
var tileServiceUrl  ="/rest/TileService/defaultService";
var baseLayer = W.mspTileLayer(tileServiceUrl).addTo(map);
```

构造函数

工厂	描述
W.mspTileLayer(String tileServiceUrl,<Options options> options)	

选项

选项名	类型	默认值	描述
minZoom	Number	0	最小显示级别
maxZoom	Number	18	最大显示级别
key	String		访问服务的凭证

FeatureLayer

继承自：Layer

显示GeoJSON要素 。要素如果未指定样式，会使用默认样式。 具体支持属性请见： Marker、Polyline、Polygon

示例

```
var feature = {
    type:"Feature",
    geometry:{
       "type": "Point",
       "coordinates": [100.0, 0.0]
    },
    properties:{

    }
}

var layer = W.featureLayer().addTo(map);
layer.addFeature(feature);
```

构造函数

工厂	描述
W.featureLayer(<Options options> options)	

选项

选项名	类型	默认值	描述
defaultStyle	Object	null	要素的默认样式，详见：Marker、Polyline、Polygon

事件

事件	数据	描述
featureclick	FeatureEvent	左键点击要素时触发
featuredblclick	FeatureEvent	左键双击要素时触发
featuremouseover	FeatureEvent	鼠标进入要素时触发
featuremouseout	FeatureEvent	鼠标离开要素时触发
featurecontextmenu	FeatureEvent	右键点击要素时触发
featurepopupopen	FeatureEvent	打开气泡时触发
featurepopupclose	FeatureEvent	关闭气泡时触发

方法

方法	返回值	描述
addFeature(Object feature,Options option)	this	添加一个要素，如果添加的要素没有id，会自动生成一个id，但不会修改传入的要素对象
addFeatures(Object[] features,Options option)	this	添加一组要素
removeFeature(String id)	this	移除要素
clearFeatures()	this	移除所有要素
getFeatures()	Object[]	获取所有要素
updateFeature(String id,Object newFeature,Options options)	this	更新要素
getFeature(String id)	Object	根据id获取要素，
getFeatureId(Object feature)	String	获取要素的ID
openPopup(String id,String content,Object options)	this	在指定要素上打开一个气泡
closePopup()	this	关闭气泡
setStyle(String id,Options styleOptions)	this	设置某个要素的样式
setDefaultStyle(Object style)	this	设置整体图层的默认样式
enableEdit(String id)	this	让某个要素可编辑
enableEdit()	this	让所有要素可编辑
disableEdit(String id)	this	禁用指定要素的编辑，禁用之前会先保存编辑效果
disableEdit()	this	禁用所有要素的编辑，禁用之前会先保存编辑效果
commitEdit(String id)	this	保存指定要素的编辑
commitEdit()	this	保存所有要素的编辑
rollbackEdit(String id)	this	回滚指定要素的编辑
rollbackEdit()	this	回滚所有要素的编辑
getCenter(String id)	Point	获取指定要素的中心点
getFeatureBounds(String id)	Bounds	获取指定要素的范围
getBounds()	Bounds	获取FeatureLayer的范围

Marker

继承自Layer。

在地图上显示可点击或拖拽的图标。

示例

```
W.marker([50.5, 30.5]).addTo(map);
```

构造函数

方法	描述
W.marker(<LatLng >latlng,< Marker options > options?)	

选项

选项	类型	默认值	描述
icon	Icon	null	Icon类用来指定marker的绘制方式，如果未指定
title	String	" "	当滑过marker时，浏览器的提示文字
zIndexOffset	Number	0	默认情况下，marker图片的zIndex自动根据经纬度计算。指定一个很大的值（如1000），可使控制marker显示在其他东西的上面，如果指定一个很大的负值，则会在其他marker的下面。
opacity	Number	1.0	不透明度
riseOnHover	Boolean	false	如果为true,当鼠标滑过Marker时，marker会显示到顶部
riseOffset	Number	250	riseOnHover启用后的zIndex偏移量

方法

方法	返回值	描述
toGeoJSON()	Object	获取GeoJSON格式描述 (GeoJSON点要素)
getLatLng()	LatLng	获取当前位置
setLatLng(<LatLng> latLng)	this	设置当前位置.
setZIndexOffset(< Number > offset)	this	设置zIndex偏移
setIcon(<Icon> icon)	this	设置icon.
setOpacity(< Number> opacity)	this	设置不透明度.
enableEdit()	this	启用编辑
disableEdit()	this	禁止编辑

CircleMarker

继承自Layer。

表示一个有固定半径(单位：像素)的圆。

构造

工厂	描述
W.circleMarker(< LatLng> latlng, < CircleMarker options> options?)	latlng是中心点

选项

选项	类型	默认值	描述
stroke	Boolean	true	是否绘制边框.
color	String	'#3388ff'	边框颜色
weight	Number	3	边框粗细，单位是像素
opacity	Number	1.0	边框不透明度
lineCap	String	'round'	两端形状 请参照
lineJoin	String	'round'	拐角处形状， 请参照
fill	Boolean		是否有填充效果
fillColor	String	*	填充颜色，默认和color相同
fillOpacity	Number	0.2	填充不透明度
fillRule	String	'evenodd'	填充规则，请参照
className	String	null	自定义样式
radius	Number	10	圆的半径，单位是像素

方法

方法	返回值	描述
toGeoJSON()	Object	获取GeoJSON描述 (Point Feature).
setLatLng(<LatLng> latLng)	this	设置位置
getLatLng()	LatLng	获取位置
setRadius(<Number> radius)	this	设置半径，单位是像素
getRadius()	Number	获取半径，单位是像素

Circle

继承自Layer。

在地图上绘制圆。它是一个近似圆形，在接近两极的地方会发生变化（因为投影失真）。

示例

```
W.circle([50.5, 30.5], {radius: 200}).addTo(map);
```

构造

工厂	描述
W.circle(<LatLng> latlng, <Circle options> options?)	latlng是圆心，options中包括半径

选项

选项	类型	默认值	描述
stroke	Boolean	true	是否绘制边框.
color	String	'#3388ff'	边框颜色
weight	Number	3	边框粗细，单位是像素
opacity	Number	1.0	边框不透明度
lineCap	String	'round'	两端形状 请参照
lineJoin	String	'round'	拐角处形状， 请参照
fill	Boolean		是否有填充效果
fillColor	String	*	填充颜色，默认和color相同
fillOpacity	Number	0.2	填充不透明度
fillRule	String	'evenodd'	填充规则，请参照
className	String	null	自定义样式
radius	Number		圆的半径，单位是米

方法

方法	返回值	描述
setRadius(< Number> radius)	this	设置半径，单位是米
getRadius()	Number	获取半径，单位是米
getBounds()	LatLngBounds	获取矩形边界

enableEdit()	this	启用编辑
disableEdit()	this	禁止编辑

Polyline

继承自Layer。

用来在地图上绘制折线。

示例

```
// 使用坐标数组创建一条红色的折线 。
var latlngs = [
    [45.51, -122.68],
    [37.77, -122.43],
    [34.04, -118.2]
];
var polyline = W.polyline(latlngs, {color: 'red'}).addTo(map);

map.fitBounds(polyline.getBounds());
```

也可以使用多维数组表示一个多线

```
var latlngs = [
    [[45.51, -122.68],
     [37.77, -122.43],
     [34.04, -118.2]],
    [[40.78, -73.91],
     [41.83, -87.62],
     [32.76, -96.72]]
];
```

构造

工厂	描述
W.polyline(< LatLng[]> latlngs, < Polyline options> options?)	使用坐标数组创建折线，也可以使用多维数组创建多线

选项

选项	类型	默认值	描述
stroke	Boolean	true	是否绘制边框.
color	String	'#3388ff'	边框颜色
weight	Number	3	边框粗细，单位是像素
opacity	Number	1.0	边框不透明度
lineCap	String	'round'	两端形状 请参照
lineJoin	String	'round'	拐角处形状， 请参照
fill	Boolean		是否有填充效果

fillColor	String	*	填充颜色，默认和color相同
fillOpacity	Number	0.2	填充不透明度
fillRule	String	'evenodd'	填充规则，请参照
className	String	null	自定义样式
smoothFactor	Number	1.0	控制不同显示级别抽稀程度。越大表示更好的性能和更加平滑。越小则越精确

方法

方法	返回值	描述
toGeoJSON()	Object	返回GeoJSON格式，（可能是LineString Feature或MultiLineString Feature)
getLatLngs()	LatLng[]	返回坐标数组
setLatLngs(<LatLng[]> latlngs)	this	设置坐标
isEmpty()	Boolean	如果没有坐标，则返回true
getCenter()	LatLng	获取中心点
getBounds()	LatLngBounds	获取边界
addLatLng(<LatLng> latlng)	this	添加坐标点
enableEdit()	this	启用编辑
disableEdit()	this	禁止编辑

Polygon

继承自Layer。 用来在地图上绘制多边形。请注意，使用坐标数组创建Polygon时，不需要在最后追加和第一个点一样的坐标。

示例

```
// 使用坐标数组创建一个红色的多边形
var latlngs = [[37, -109.05],[41, -109.03],[41, -102.05],[37, -102.04]];
var polygon = W.polygon(latlngs, {color: 'red'}).addTo(map);
// 缩放到合适的级别
map.fitBounds(polygon.getBounds());
```

还可以数组的数组。第一个数组表示多边形的外边框，剩下的数组表示边框内部的洞。

```
var latlngs = [
  [[37, -109.05],[41, -109.03],[41, -102.05],[37, -102.04]], // outer ring
  [[37.29, -108.58],[40.71, -108.58],[40.71, -102.50],[37.29, -102.50]] // hole
];
```

你还可以使用多维数组表示MultiPolygon 形状。

```
var latlngs = [
  [ // 第1个 polygon
    [[37, -109.05],[41, -109.03],[41, -102.05],[37, -102.04]], // outer ring
    [[37.29, -108.58],[40.71, -108.58],[40.71, -102.50],[37.29, -102.50]] // hole
  ],
  [ // 第2个 polygon
    [[41, -111.03],[45, -111.04],[45, -104.05],[41, -104.05]]
  ]
];
```

构造

工厂	描述
W.polygon(<LatLng[]> latlngs, <Polygon options> options?)	

选项

选项	类型	默认值	描述
stroke	Boolean	true	是否绘制边框.
color	String	'#3388ff'	边框颜色
weight	Number	3	边框粗细，单位是像素
opacity	Number	1.0	边框不透明度
lineCap	String	'round'	两端形状 请参照

lineJoin	String	'round'	拐角处形状， 请参照
fill	Boolean		是否有填充效果
fillColor	String	*	填充颜色，默认和color相同
fillOpacity	Number	0.2	填充不透明度
fillRule	String	'evenodd'	填充规则，请参照
className	String	null	自定义样式
smoothFactor	Number	1.0	控制不同显示级别稠稀程度。越大表示更好的性能和更加平滑。越小则越精确

方法

方法	返回值	描述
toGeoJSON()	Object	返回GeoJSON格式， (可能是Polygon Feature或 MultiPolygon Feature)
enableEdit()	this	启用编辑
disableEdit()	this	禁止编辑

Icon

描述一个图标，用来创建Marker。

示例

```
var myIcon = W.icon({
    iconUrl: 'my-icon.png',
    iconSize: [38, 95],
    iconAnchor: [22, 94],
    popupAnchor: [-3, -76],
    shadowUrl: 'my-icon-shadow.png',
    shadowSize: [68, 95],
    shadowAnchor: [22, 94]
});
W.marker([50.505, 30.57], {icon: myIcon}).addTo(map);
```

构造函数

工厂	描述
W.icon(< Icon options > options)	

构造选项

选项

选项名	类型	默认值	描述
iconUrl	String	null	图片url
iconSize	Point	null	图片尺寸，单位为像素
iconAnchor	Point	null	图标上对应真正地理位置的点，单位是像素
popupAnchor	Point	null	指定弹出气泡的位置
shadowUrl	String	null	阴影图片，单位是像素
shadowSize	Point	null	阴影的尺寸，单位是像素
shadowAnchor	Point	null	阴影上对应真正地理位置的点，单位是像素
className	String	" "	自定义类名

DivIcon

类似Icon,但是使用DIV，代理Image，用来创建Marker。

示例

```
var myIcon = W.divIcon({className: 'my-div-icon'});
 W.marker([50.505, 30.57], {icon: myIcon}).addTo(map);
```

构造函数

工厂	描述
W.divIcon(< DivIcon options > options)	

构造选项

选项

选项名	类型	默认值	描述
html	String	" "	内容
bgPos	Point	[0, 0]	相对于背景的位置
iconSize	Point	null	图片尺寸，单位为像素
iconAnchor	Point	null	图标上对应真正地理位置的点，单位是像素
popupAnchor	Point	null	指定弹出气泡的位置
className	String	" "	自定义类名

LayerGroup

继承自Layer。 将多个图层作为一组，当成一个整体进行处理。将LayerGroup添加到地图上，从组中添加和删除图层时，会自动的从map中添加和删除。

示例

```
W.layerGroup([marker1, marker2])
    .addLayer(polyline)
    .addTo(map);
```

构造

工厂	描述
W.layerGroup(< Layer[]> layers)	layers 是初始包含的图层

方法

方法	返回值	描述
addLayer(< Layer> layer)	this	向组中添加图层
removeLayer(< Layer> layer)	this	从组中移除图层
removeLayer(< Number> id)	this	根据内部id移除图层
hasLayer(< Layer> layer)	Boolean	判断当前group中是否包含该指定图层
clearLayers()	this	清空组中所有图层
invoke(< String> methodName, …)	this	调用组中每个图层上指定的方法
eachLayer(< Function> fn, context?)	this	遍历组中所有图层 ， 可选的context是函数执行的上下文
getLayers()	Layer[]	获取所有图层
setZIndex(<Number> zIndex)	this	设置所有图层的z-Index

Popup

用来在地图特定位置上弹出气泡。如果同一时刻只显示一个气泡，使用Map.openPopup()。

示例

在点击一个marker时弹出气泡

```
marker.bindPopup(popupContent).openPopup();
```

Poth Overlay也有bindPopuup方法。下面是一重更复杂的方式弹出气泡。

```
var popup = W.popup()
    .setLatLng(latlng)
    .setContent('<p>Hello world!<br />This is a nice popup.</p>')
    .openOn(map);
```

构造

方法	描述
W.popup(< Popup options > options?,< Layer > source?)	options可设置外观、位置。 可选的souce 用来指定相关的图层

选项

选项	类型	默认值	描述
maxWidth	Number	300	弹出气泡最大宽度, 单位是像素
minWidth	Number	50	弹出气泡最小宽度, 单位是像素
maxHeight	Number	null	如果指定的该值，当内容过多时，会出现滚动条
autoPan	Boolean	true	如果不想在弹出气泡时出现移动动画，将该值设置为false
autoPanPaddingTopLeft	Point	null	自动平移地图后，气泡离左上角的距离
autoPanPaddingBottomRight	Point	null	自动平移地图后，气泡到右下角的距离
autoPanPadding	Point	Point(5, 5)	同时设置自动平移后，气泡到左上角及右下角的距离
keepInView	Boolean	false	设置为true之后，可以防止通过移动地图导致气泡在屏幕之外
closeButton	Boolean	true	在气泡中显示关闭按钮
closeOnClick	Boolean	*	设置在点击地图时，气泡是否自动关闭。默认值取Map的closePopupOnClick选项。
			默认情况下其他气泡打开时，该气泡会自动关

autoClose	Boolean	true	闭，如果不想要这种默认行为，将该值设置为 false
className	String	''	给气泡设置自定义css类

方法

方法	返回值	描述
getLatLng()	LatLng	获取气泡所在位置
setLatLng(<LatLng> latlng)	this	设置气泡弹出位置
getContent()	String/HTMLElement	气泡内容元素
setContent(< String / HTMLElement / Function > htmlContent)	this	设置气泡内容，如果传入的是一个函数，设置该函数时会传入source layer，执行该函数后必须返回一个String或者HTMLElement
getElement()	String / HTMLElement	同 getContent()
update()	null	更新气泡的内容、位置，当气泡内置的某些东西发生变化（如图片已加载）时，调用该方法非常有用。
isOpen()	Boolean	判断该气泡是否在地图显示
bringToFront()	this	将该气泡放置到其他气泡前面 (相同的地图面板)
bringToBack()	this	将该气泡放置到其他气泡后面 (相同的地图面板)
openOn(<Map> map)	this	将气泡添加到地图上，并且关闭之前的气泡，同 map.openPopup(popup)

Control

所有控件类的父类

构造函数

选项

选项名	类型	默认值	描述
position	String	'topright'	控件的位置，可取'topleft'、 'topright'、 'bottomleft'、 'bottomright'。

方法

方法名	返回值	描述
addTo(< Map > type)	this	将图层添加到地图中
remove()	this	从地图中移除图层

Control.Zoom

继承自Control。

缩放控件

构造

工厂	描述
W.control.zoom()	创建一个缩放控件

Control.Attribution

继承自Control

属性控件允许你在地图上一个文本框里显示属性数据。

构造

工厂	描述
W.control.attribution(< Control.Attribution options >)	创建一个属性控件。

选项

选项名	类型	默认值	描述
prefix	String	'wmap'	属性前面显示的HTML文本。通过false禁用。

方法

方法名	返回值	描述
setPrefix (String prefix)	this	设置属性前的文本。
addAttribution (String text)	this	添加属性文本(例如'Vector data © wmap')。
removeAttribution (String text)	this	删除属性文本。

Control.Layers

继承自Control

图层控件让用户能够在不同的层之间切换。

示例

```
var baseLayers = {
    "Mapbox": mapbox,
    "OpenStreetMap": osm
};
var overlays = {
    "Marker": marker,
    "Roads": roadsLayer
};
W.control.layers(baseLayers, overlays).addTo(map);
```

baseLayers和overlays参数是以图层名称为键值和图层对象作为值的对象值：

```
{
    "<someName1>": layer1,
    "<someName2>": layer2
}
```

图层名称可以包含HTML，这样允许你可以向项目添加其他样式：

```
{"<img src='my-layer-icon' /><span class='my-layer-item'>My Layer</span>": myLayer}
```

构造

工厂	描述
W.control.layers ( )	

方法

方法名	返回值	描述
addBaseLayer(< Layer> layer;< String > name)	this	向控件添加具有指定名称的基本图层（单选按钮条目）
addOverlay(< Layer> layer;< String > name)	this	向控件添加具有给定名称的叠加图层（复选框条目）
removeLayer(< Layer> layer)	this	从控件中删除指定的图层。

Control.Scale

继承自Control

比例尺控件，以公制（m / km）显示当前屏幕中心的比例

```
W.control.scale().addTo(map);
```

构造

工厂	描述
W.control.scale()	创建比例尺控件。

Control.MiniMap

继承自Control 迷你地图控件。

构造

工厂	描述
W.control.minimap(< Layer > layer)	创建迷你地图，显示指定的图层。

LatLng

描述一个经纬度地理位置

示例

```
var latlng = W.latLng(50.5, 30.5);
```

所有接受LatLng实例的地方，都可接受一个数组或者一个对象（除非特别声明），所以下面的这些代码是等价的：

```
map.panTo([50, 30]);
map.panTo(W.latLng(50, 30));
```

构造函数

方法名	描述
W.latLng(< Number > latitude, < Number > longitude)	指定纬度和经度
W.latLng(< Array > coords)	[纬度,经度]

方法

方法	返回值	描述
equals(< LatLng > otherLatLng, < Number > maxMargin?)	Boolean	判断两个位置是否相等，maxMargin设置允许的误差
distanceTo(< LatLng > otherLatLng)	Number	计算到另一个位置的距离，单位是米

属性

属性	类型	描述
lat	Number	纬度
lng	Number	经度

LatLngBounds

描述地图上的一个长方形区域，单位是经纬度

示例

```
var corner1 = W.latLng(40.712, -74.227),
corner2 = W.latLng(40.774, -74.125),
bounds = W.latLngBounds(corner1, corner2);
```

所有接受 LatLngBounds 实例的地方，都可接受一个数组（除非特别声明）,如：

```
map.fitBounds([
    [40.712, -74.227],
    [40.774, -74.125]
]);
```

构造函数

方法名	描述
W.latLngBounds(<LatLng> corner1, <LatLng> corner2))	使用对角两个点，创建LatLngBounds
W.latLngBounds(<LatLng[]> latlngs))	使用对角两个点，创建LatLngBounds

方法

方法	返回值	描述
extend(<LatLng> latlng)	this	扩展范围，包含指定的点
extend(< LatLngBounds > otherBounds)	this	扩展范围，包含指定的区域
pad(< Number > bufferRatio)	LatLng / LatLngBounds	按百分比扩展
getCenter()	LatLng	返回中心点
getSouthWest()	LatLng	返回西南方位置
getNorthEast()	LatLng	返回东北方位置
getNorthWest()	LatLng	返回西北方位置
getSouthEast()	LatLng	返回东南方位置
contains(<LatLngBounds> otherBounds)	Boolean	是否包含指定的区域
contains(<LatLng> latlng))	Boolean	是否包含指定的点
intersects(<LatLngBounds>		比较两个区域是否相交（至少有一个交

otherBounds)		点）
overlaps(<Bounds> otherBounds)	Boolean	比较两个区域是否重叠（有共同的区域）
equals(<LatLngBounds>	Boolean	比较两个区域是否相等

Point

描述屏幕上的一个点，单位是像素

示例

```
var point = W.point(200, 300);
```

所有接受Point实例的地方，都可接受一个数组，所以下面的这些代码是等价的：

```
map.panBy([200, 300]);
map.panBy(W.point(200, 300));
```

构造函数

方法名	描述
W.point(< Number > x, < Number > y)	
W.point(< Number []> coords)	[x,y]

属性

属性	类型	描述
x	Number	x坐标
y	Number	y坐标

Bounds

描述屏幕上的一个长方形区域，单位是像素

示例

```
var p1 = W.point(10, 10),
p2 = W.point(40, 60),
bounds = W.bounds(p1, p2);
```

所有接受 Bounds 实例的地方，都可接受一个数组（除非特别声明）,如：

```
otherBounds.intersects([[10, 10], [40, 60]]);
```

构造函数

构造

方法名	描述
W.latLngBounds(<Point> topLeft, <Point> bottomRight))	指定左上角和右下角
W.latLngBounds(<Point[]> points))	

方法

方法	返回值	描述
extend(<Point> point)	this	扩展范围，包含指定的点
getSize()	Point	返回的point的中x表示长度，y表示高度
getCenter()	Point	返回中心点
getBottomLeft()	Point	左下角
getTopRight()	Point	右上角
contains(<Bounds> otherBounds)	Boolean	如果包含指定的区域，返回true
contains(<Point> point)	Boolean	如果包含指定的点，返回true
intersects(<Bounds> otherBounds)	Boolean	如果两个区域相交(至少有一个共同点），返回true
overlaps(<Bounds> otherBounds)	Boolean	如果两个区域重叠（共同区域是一个面，而不是一个点），返回true

属性

属性	类型	描述
min	Point	左上角
max	Point	右下角

Event

所有事件类的父类。

属性

属性	类型	描述
type	String	事件类型
target	Object	事件源

KeyboardEvent

继承Event

属性

属性	类型	描述
originalEvent	DOMEvent	原始的浏览器事件

MouseEvent

继承Event

属性

属性	类型	描述
latlng	LatLng	地理位置
layerPoint	Point	相对于所在layer的像素位置
containerPoint	Point	相对于地图容器的像素位置
originalEvent	DOMEvent	原始的浏览器事件

LayerEvent

继承Event

属性

属性	类型	描述
layer	Layer	被添加或删除的图层

LayerControlEvent

继承Event

属性

属性	类型	描述
layer	Layer	被添加或删除的图层
name	String	图层名称

ResizeEvent

继承Event

属性

属性	类型	描述
oldSize	Point	旧尺寸
newSize	Point	新尺寸

PopupEvent

继承Event

属性

属性	类型	描述
popup	PopupPopup	打开或关闭的气泡

FeatureEvent

继承Event

属性

属性	类型	描述
feature	Object	发生事件的要素对象

Evented

提供事件相关方法。

方法

方法名	返回值	描述
on(< String > type,< Function> fn,< Object > context?)	this	绑定事件处理方法
off(< String > type,< Function> fn,< Object > context?)	this	移除事件处理方法

config

引入config.js后，会设置地图的默认范围、最大级别和最小级别

示例

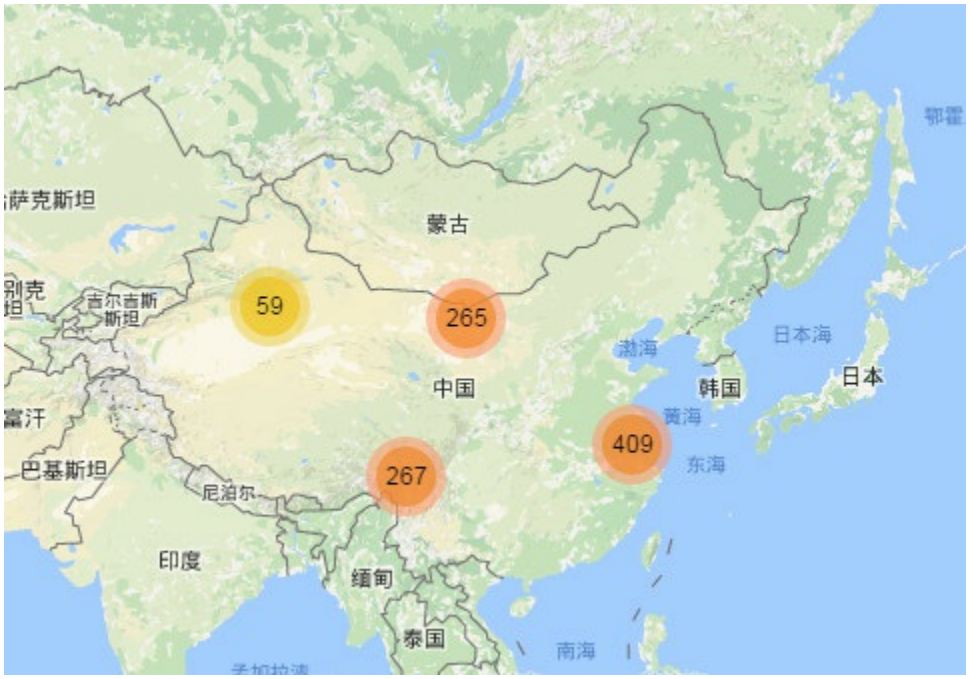
```
map.setCenter(W.config.getDefaultCenter())
```

方法

方法名	返回值	描述
getMaxBounds()	LatLngBounds	获取地图的最大范围
getMinZoom()	LatLngBounds	获取地图最小级别
getMaxZoom()	LatLngBounds	获取地图最大级别
getDefaultZoom()	LatLngBounds	获取地图默认级别
getDefaultCenter()	LatLngBounds	获取地图默认中心点

聚合图组件

效果图



用法

示例

```
//创建地图，并设置初始中心点和级别
map = W.map("mapContainer", {
    center : [35.53222622770337, 106.875],
    zoom : 5,
    serverUrl: 'http://localhost:8080/tileProxy'
});
//创建聚合图标注组
var markers = new W.MarkerClusterGroup();
var markersList = [];
//生成固定个坐标
function populate() {
    for (var i = 0; i < 1000; i++) {
        var m = new W.Marker(getRandomLatLng(map));
        markersList.push(m);
        markers.addLayer(m);
    }
    return false;
}
//在屏幕范围内随机生成一个坐标
function getRandomLatLng(map) {
    var bounds = map.getBounds(),
    southWest = bounds.getSouthWest(),
    northEast = bounds.getNorthEast(),
```

```
    lngSpan = northEast.lng - southWest.lng,
    latSpan = northEast.lat - southWest.lat;
    return new W.LatLng(
        southWest.lat + latSpan * Math.random(),
        southWest.lng + lngSpan * Math.random());
}
markers.on('clusterclick', function (a) {
    //console.log(a);
});
markers.on('click', function (a) {
    //alert('marker ' + a.layer);
});
populate();
//将聚合图添加到地图上
map.addLayer(markers);
```

构造方法

构造方法	描述
W.MarkerClusterGroup(opt?)	构造方法

构造属性

属性名	类型	默认值	描述
showCoverageOnHover	boolean	true	当鼠标经过聚合图标时，是否显示所有被聚合标注的边框范围。
zoomToBoundsOnClick	boolean	true	当点击聚合图标时，是否切换地图当前可视范围到聚合标注范围。
spiderfyOnMaxZoom	boolean	true	当地图在最大级别时，点击聚合图标是否显示所有被聚合的标注。
removeOutsideVisibleBounds	boolean	true	是否显示被聚合标注的外边框。
spiderLegPolylineOptions	object	{ weight: 1.5, color: '#222', opacity: 0.5 }	当鼠标放置在聚合图标上时，显示的范围线的样式信息。
iconCreateFunction	function	无	用于自定义marker。

方法

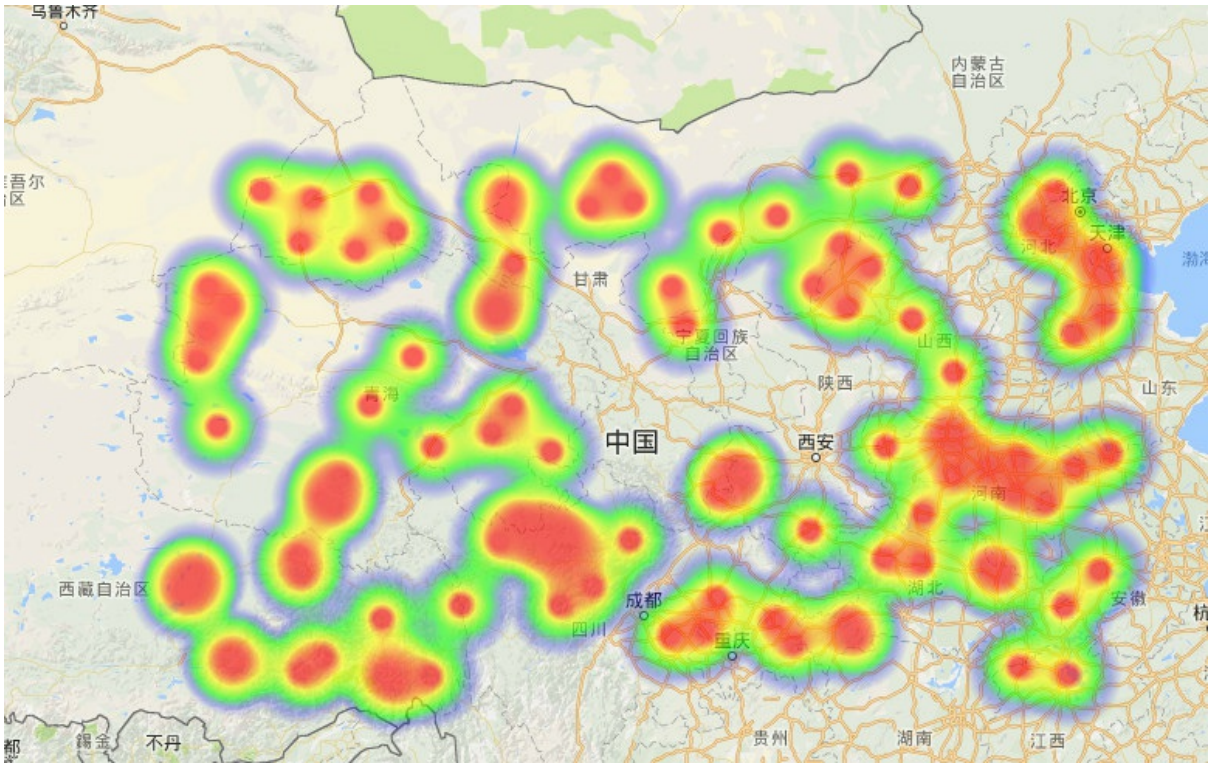
方法名称	描述
addLayer(W.Marker)	添加一个标注到聚合图层。
addLayers(W.Marker[])	添加一组标注到聚合图层。
removeLayer(W.Marker)	从聚合图层删除一个标注。
removeLayers(W.Marker[])	从聚合图层删除一组标注。
clearLayers()	清空标注图层。

事件

事件名称	用法描述
clusterclick	`markers.on('clusterclick',function(a){// a.layer 为聚合图对象 console.log('cluster '+ a.layer.getAllChildMarkers().length);});`

热力图组件

效果图



示例

```
<script type="text/javascript">
        var map, baseLayer, heatMaplayer;
        $(document).ready(function(){
            initMap();
        });
      function initMap(){
          baseLayer = new W.tileLayer('http://localhost:8080/tileProxy/getImage?x={x}&y={y}&z={z}&type=google');
          map = W.map("mapContainer", {
              zoom : 5,
              center : [35.53222622770337, 106.875],
              maxZoom : 11,
              layers: [baseLayer]
          });
          initHeatMap();
      }
      function initHeatMap(data){
          var _data = [];
          //在指定坐标范围内生成随机点坐标
          var minX = 90.7470703125;
          var maxX = 117.333984375;
          var minY = 28.8831596093235;
          var maxY = 40.81380923056958;
          var _x = maxX - minX;
```

```
            var _y = maxY - minY;
            for(var i = 0; i < 100; i++){
                var x = minX + Math.random() * _x;
                var y = minY + Math.random() * _y;

                _data.push({
                    lat: y,
                    lng: x,
                    count: 1
                });
            }
            var testData = {
                max: 1,
                data: _data
            };
            var cfg = {
                //半径 单位像素
                "radius": 35,
                //最大透明度
                "maxOpacity": .6,
                //是否根据级别变化进行缩放
                "scaleRadius": false,
                lngField: 'lng',//经度字段名称
                latField: 'lat',//纬度字段名称
                valueFiled : 'count'//用于生成热图的值字段
            };
            heatMaplayer = new HeatmapOverlay(cfg);

            map.addLayer(heatMaplayer);
            heatMaplayer.setData(testData);
        }
</script>
```

构造方法

构造方法	描述
HeatmapOverlay(opt)	构造方法

构造属性

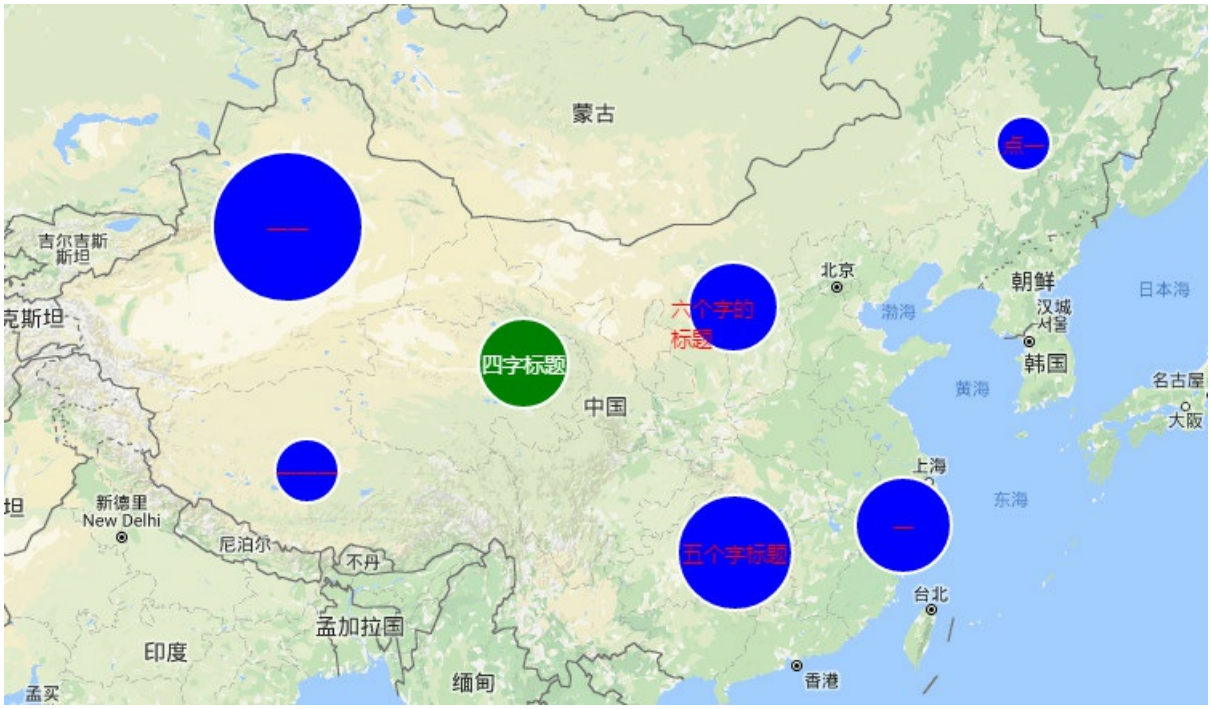
属性名	类型	默认值	描述
radius	number	无	颜色点半径。
opacity	number[0, 1]	0.6	透明度。
maxOpacity	number[0, 1]	1	最大透明度
scaleRadius	boolean	true	是否随地图放大缩小而改变半径。
lngField	string	无	数据中代表经度的字段。
latField	string	无	数据中代表纬度的字段。
valueFiled	string	无	数据中代表值的字段，该字段用于生成热度。

方法

方法名称	描述
setData(object[])	向热力图图层中增加数据。

符号图组件

效果图



示例

```
        //定义样式
        var style = {
            weight : 2,
            opacity : 1,
            fillOpacity : 1,
            color : 'white',
            fillColor : 'blue',
            fontSize: 13,
            fontColor: 'red'
        };

        //鼠标选中时的样式
        var selectStyle = {
            weight: 2,
            color: 'white',
            dashArray: '',
            fillColor : 'green',
            fillOpacity: 1,
            fontSize: 14,
            fontColor: 'white'
        };
        var opt = {
            style : style,
            selectStyle : selectStyle
        }
    //生产符号图的数据结构
```

```
var data = [{
    label : '一',
    latlng : [29.169763146098717, 120.04928890172152],
    radius : 30,
    popupContent : "气泡内容",
    data : {}
},{
    label : '一一',
    latlng : [42.35854391749705, 86.30859375],
    radius : 47,
    popupContent : "气泡内容",
    data : {}
},{
    label : '一一一',
    latlng : [31.765537409484374, 87.36328125],
    radius : 20,
    popupContent : "气泡内容",
    data : {}
},{
    label : '四字标题',
    latlng : [36.56260003738548, 99.228515625],
    radius : 28,
    popupContent : "气泡内容",
    data : {}
},{
    label : '五个字标题',
    latlng : [27.839076094777816, 110.7861328125],
    radius : 36,
    popupContent : "气泡内容",
    data : {}
},{
    label : '六个字的标题',
    latlng : [38.993572058209466, 110.74218749999999],
    radius : 28,
    popupContent : "气泡内容",
    data : {}
},{
    label : '点一',
    latlng : [45.67548217560647, 126.60644531250001 ],
    radius : 17,
    popupContent : "气泡内容",
    data : {}
}];

var lyr = W.symbol(data, opt);

map.addLayer(lyr);
```

构造方法

构造方法	描述
W.Symbol(opt)	构造方法

构造属性

属性名	类型	默认值	描述
style	object	{ weight : 2, opacity : 1, fillOpacity : 1, color : 'white', fillColor : 'red', fontSize: 13, fontColor: 'white' }	符号图叠加到地图上时的默认样式。

selectStyle	object	{ weight: 2, color: 'white', dashArray: '', fillColor : 'green', fillOpacity: 1, fontSize: 14, fontColor: 'red' }	符号图在鼠标划过时的样式。
callback	object	{ click : function(data, marker){ } }	回调事件集合，当前只支持click回调事件。

用法

```
//创建图表层
var symbolLayer = W.symbol(data);
//将图表层添加到地图上
map.addLayer(symbolLayer);
```

图表组件

图表插件基于echarts3.x版本实现，所以图表属性相关的参数可以直接参考echarts

效果图

示例

```
var data = [{
    latlng : [39.095962936305504, 91.23046875],
    height: 100,
    width: 150,
    chartOption : {//chartOption 这部分属性可以参考echarts3.x的图表属性文档
        tooltip : {
            show: true
        },
        axis : {
            axisLine : false
        },
        xAxis : [
        {
            type : 'category',
            data : ['煤炭','钢铁','石油']
        }
        ],
        yAxis : [
        {
            type : 'value'
        }
        ],
```

```
            series : [
            {
                name:'产量',
                type:'bar',
                data:[2.0, 4.9, 7.0]
            }
            ],
            grid : {
                borderColor : '#ffffff',
                x : 20,
                y : 15,
                x2 : 15,
                y2 : 30
            }
        }
    },{
        latlng : [27.059125784374068, 111.884765625],
        height: 100,
        width: 150,
        chartOption : {
            tooltip : {
                show: true
            },
            axis : {
                axisLine : false
            },
            xAxis : [
            {
                type : 'category',
                data : ['煤炭','钢铁','石油']
            }
            ],
            yAxis : [
            {
                type : 'value'
            }
            ],
            series : [
            {
                name:'产量',
                type:'bar',
                data:[2.0, 4.9, 7.0]
            }
            ],
            grid : {
                borderColor : '#ffffff',
                x : 20,
                y : 15,
                x2 : 15,
                y2 : 30
            }
        }
    },{
        latlng : [45.213003555993964, 123.662109375],
        height: 100,
        width: 150,
        chartOption : {
            tooltip : {
                show: true
            },
            axis : {
                axisLine : false
            },
            xAxis : [
            {
                type : 'category',
```

```
                data : ['煤炭','钢铁','石油']
            }
            ],
            yAxis : [
            {
                type : 'value'
            }
            ],
            series : [
            {
                name:'产量',
                type:'bar',
                data:[2.0, 4.9, 7.0]
            }
            ],
            grid : {
                borderColor : '#ffffff',
                x : 20,
                y : 15,
                x2 : 15,
                y2 : 30
            }
        }
    }];
    map.addLayer(W.charts(data, {
        callback: {
            click : function(marker){
                alert(marker.mId);
            }
        }
    }));
```

构造方法

构造方法	描述
W.charts(data, options)	构造方法 data: 为一个object 数组； options: 为构造属性；

构造属性

属性名	类型	默认值	描述
defaultIconWidth	number	240	图表默认宽度。
defaultIconHeight	number	120	图表默认高度。
callback	object	{ click: function(marker){ }}	当图标被点击时的事件

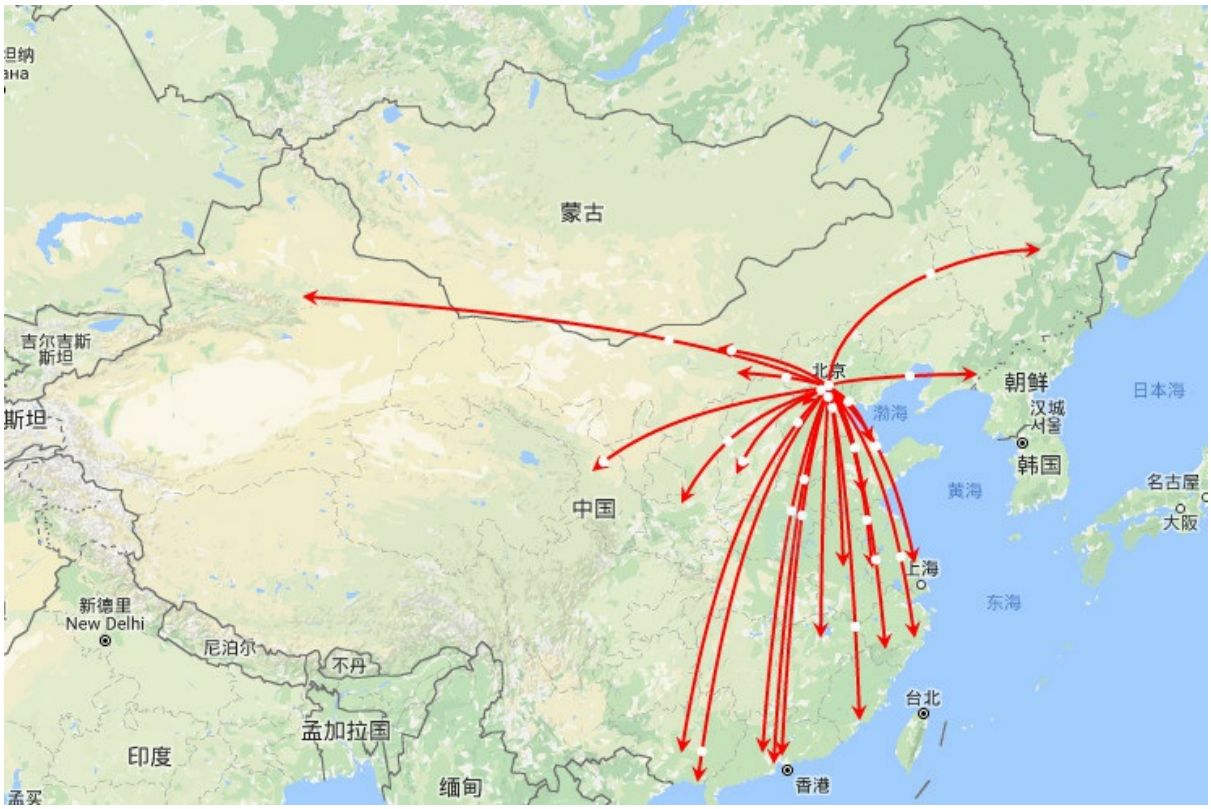
数据格式

```
{
    latlng : [39.095962936305504, 91.23046875], //必须
    height: 100, //图表高度
    width: 150, //图表宽度
    chartOption : { // charts 属性可以参考echarts2.x 版本api
        tooltip : {
            show: true
        },
```

```
        axis : {
            axisLine : false
        },
        xAxis : [
        {
            type : 'category',
            data : ['煤炭','钢铁','石油']
        }
        ],
        yAxis : [
        {
            type : 'value'
        }
        ],
        series : [
        {
            name:'产量',
            type:'bar',
            data:[2.0, 4.9, 7.0]
        }
        ],
        grid : {
            borderColor : '#ffffff',
            x : 20,
            y : 15,
            x2 : 15,
            y2 : 30
        }
    }
    }
```

流向图组件

效果图



用法示例

```
<script type="text/javascript">
   var map,baseLayer,flow;
   var city = {
     '东莞': [113.8953,22.901],
     '东营': [118.7073,37.5513],
     '中山': [113.4229,22.478],
     '临汾': [111.4783,36.1615],
     '临沂': [118.3118,35.2936],
     '丹东': [124.541,40.4242],
     '丽水': [119.5642,28.1854],
     '乌鲁木齐': [87.9236,43.5883],
     '佛山': [112.8955,23.1097],
     '保定': [115.0488,39.0948],
     '兰州': [103.5901,36.3043],
     '包头': [110.3467,41.4899],
     '北京': [116.4551,40.2539],
     '北海': [109.314,21.6211],
     '南京': [118.8062,31.9208],
     '南宁': [108.479,23.1152],
     '南昌': [116.0046,28.6633],
     '南通': [121.1023,32.1625],
```

```
    '厦门': [118.1689,24.6478],
    '台州': [121.1353,28.6688],
    '合肥': [117.29,32.0581],
    '呼和浩特': [111.4124,40.4901],
    '咸阳': [108.4131,34.8706],
    '哈尔滨': [127.9688,45.368]
  };
  function addFlowLayer(){
   if(flow){
     map.removeLayer(flow);
   }
   var data = new Array();
   var bjLatlng = [39.90973623453719, 116.4111328125];
   for(var key in city){
     data.push({//这里构造流向图所需数据结构
       from: bjLatlng, //起点
       to: [city[key][1], city[key][0]], //终点
       color: 'red', //颜色
       size: 2,
       label: key
     });
   }
   var opt = {
     infoParse: function(data){
       return "北京    ---->     " + data.label;
     },
     callback: {
       click: function(data){
         alert(data.label);
       }
     }
   };
   flow = new W.Flow(data, opt).addTo(map);
 }
 function initMap(){
   map = W.map("mapContainer", {
    center : [35.53222622770337, 106.875],
    zoom : 5,
    serverUrl: 'http://localhost:8080/tileProxy'
  });
   addFlowLayer();
 }
 </script>
```

构造方法

构造方法	描述
W.Flow(data, options)	构造方法 data: 为一个object 数组； options: 为构造属性；

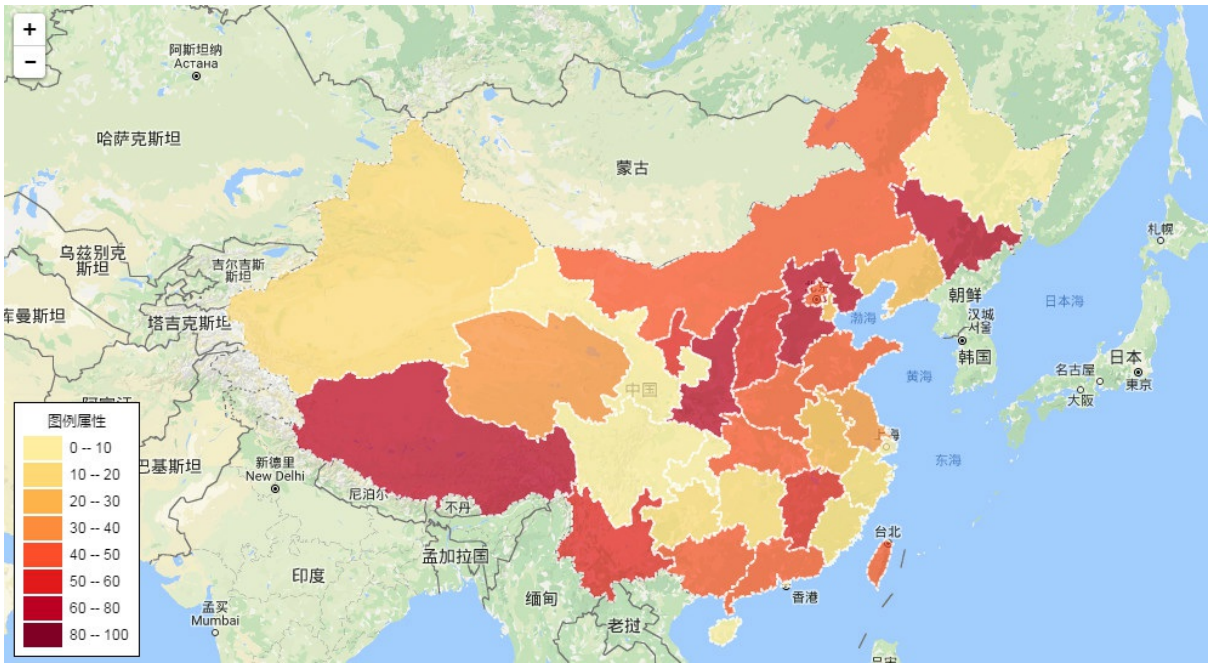
构造属性

属性名	类型	默认值	描述
style	object	{"stroke-width": 2,"stroke": "#f00","arrow-end": "classic-wide-long","stroke-dasharray": "","cursor": "pointer" }	流向线的样式。
ball	object	{ style:{ stroke: "none", fill: "#fff" }, xr: 3, yr: 3}	流向小球的样式，和大小半径。

initAnimate	boolean	true	初始化时是否需要动画。
animate	boolean	true	是否启动动画效果
showInfoWindow	boolean	true	是否显示信息窗口
infoParse	function	function(data){ return "";}	信息窗口内容转换函数
callback	object	{ click: function(data){ }, mouseover: function(data){ }, mouseout: function(data){ } }	回调事件集合

梯度图组件

效果图



示例用法

```
<script type="text/javascript">
  var map,baseLayer,info,geojson,gradeLayer;
    function initMap(){
        map = W.map("mapContainer", {
            center : [35.53222622770337, 106.875],
            zoom : 4,
    serverUrl: 'http://localhost:8080/tileProxy'
        });
      var lengeds = [
      {
          from : 0,
          to : 10,
          lable : '0 -- 10',
          color : '#FFEDA0'
      },
      {
          from : 10,
          to : 20,
          lable : '10 -- 20',
          color : '#FED976'
      },
      {
          from : 20,
          to : 30,
          lable : '20 -- 30',
          color : '#FEB24C'
      },
      {
          from : 30,
```

```
        to : 40,
        lable : '30 -- 40',
        color : '#FD8D3C'
    },
    {
        from : 40,
        to : 50,
        lable : '40 -- 50',
        color : '#FC4E2A'
    },
    {
        from : 50,
        to : 60,
        lable : '50 -- 60',
        color : '#E31A1C'
    },
    {
        from : 60,
        to : 80,
        lable : '60 -- 80',
        color : '#BD0026'
    },
    {
        from : 80,
        to : 100,
        lable : '80 -- 100',
        color : '#800026'
    }];
    var lengedOptions = {
      title : '图例属性',
      lengeds : lengeds,
      callback : {
          click : function(data, status){
              //console.log(status);
              //console.log(data);
          }
      }
};
//加载地图数据
$.getJSON("geojson/province_s_pg.geojson", function(data){
    //为测试数据增加值域
    for(var i = 0; i < data.features.length; i++){
        var feature = data.features[i];
        //像feature插入 测试数据
        feature.properties['val'] = Math.random() * 70;//值字段
    }
    gradeLayer = W.grade(data,{
        lengedOptions: lengedOptions,
        valueFiled: 'val',//值字段
        infoParse: function(data){
            var html = "信息窗内容";
            html += "<table style='font-size: 12px'>";
            for(var key in data){
                html += "<tr><td>" + key + "</td><td>" + data[key] + "</td></tr>";
            }
            html += "</table>";
            return html;
        }
    });
    map.addLayer(gradeLayer);
});
map.on("click", function(){
    if(gradeLayer){
        map.removeLayer(gradeLayer);
        gradeLayer = null;
    }else{
```

```
            //加载地图数据
            $.getJSON("geojson/province_s_pg.geojson", function(data){
                //为测试数据增加值域
                for(var i = 0; i < data.features.length; i++){
                    var feature = data.features[i];
                    //像feature插入 测试数据
                    feature.properties['val'] = Math.random() * 70;//值字段
                }
                gradeLayer = W.grade(data,{
                    lengedOptions: lengedOptions,
                    valueFiled: 'val',//值字段
                    infoParse: function(data){
                        var html = "信息窗内容";
                        html += "<table style='font-size: 12px'>";
                        for(var key in data){
                            html += "<tr><td>" + key + "</td><td>" + data[key] + "</td></tr>";
                        }
                        html += "</table>";
                        return html;
                    }
                });
                map.addLayer(gradeLayer);
            });
        }
    });
}
</script>
```

构造方法

构造方法	描述
W.Grade(data, options)	构造方法 data: 为一个object 数组； options: 为构造属性；

构造属性

属性名	类型	默认值	描述
lengedOptions	object	无	图例相关属性，参见图例组件。
valueFiled	string	null	值字段名
showInfoWindow	boolean	true	是否显示信息窗口
showLenged	boolean	true	是否显示图例
style	object	{}	样式对象
infoParse	function	function(data){ }	信息窗口内容自定义函数。

defaultStyle	object	{ //默认样式 与 path样式属性相同 weight : 2, //线宽度 opacity : 0.8, //透明度 fillOpacity : 0.7, //填充透明度 color : 'white', //线颜色 dashArray : '3', //线样式（缓冲模式） fillColor : '#004d4d' //填充色}	默认样式
selectStyle	object	{ //鼠标悬停时的样式 完整属性参照defaultStyle weight: 2, color: '#666', dashArray: '', fillOpacity: 0.7 }	选中时的样式
callback	object	{ click: function(feature){ }, mouseover: function(feature){ }, mouseout: function(feature){ }}	回调函数集合