地图API集成使用文档

作者：邓恩隆

版本：1.0　2020/11/27

后期修改人：

后期时间：

当前地图使用的中间件为superMap classic https://iclient.supermap.io/web/libs/iclient8c/apidoc/files/SuperMap/Map-js.html

它跟百度、高德不一样，它所操作的是一个图层，所以每次操作都要先生成一个对应的图层

地图上的图片如果有http或者https的资源，使用截图功能时大概率不能被canvas绘制下来，请使用本地资源

地图的初始化

数据初始化 initMapData

初始化地图 init (ele)

地图的所有操作都要先进行初始化，return 操作类的this指向，使用返回的this指向进行地图操作例如:

```
let map = new window.HM().init(eleID)
map.createdMarkCoverage("testMack")
```

使用以上方法可生成多个地图进行操作，请注意数据的共享。如果要生成一个全新数据的地图，请先new window.HM()生成数据，再使用init()加载地图

获取地图加载状态 getMapInitState() return true||false

marker

创建mark图层 createdMarkCoverage (coverageName)

- coverageName *必填 # type string 图层名字
- return promise # 通过then进行之后逻辑

清空该图层的数据 clearAtPresentMarkData (coverageName)

- coverageName *必填 # type string 图层名字
- return promise # 通过then进行之后逻辑

设置单个mark落点 setOneMarker (coverageName,data,addEventlist='')

落点之前会清除这个图层之前的点

- coverageName *必填 # string 图层名字
- data *必填 # Object 落点的数据

  longitude和latitude 作为落点依据，数据格式:

  {

  longitude:113.1,

latitude:23.1,

// imgInfo 落图标记设置

wd: 30　宽度 type Number

hg: 30 高度 type Number

src: 'imgPath'　图片路径　type String

label:'标注',

color:'' 颜色16进制

fontSize:'' 字体大小

textPosition:'' 显示位置 'tp' 顶部 'bt' 底部

}

- addEventlist　　# Object 给每个落点绑定事件的对象

  支持事件:

事件名	触发场景
click	当鼠标单击maker时触发此事件。
dblclick	当鼠标双击maker时触发此事件。
mousedown	当鼠标在maker上按下时触发此事件。
mouseup	当鼠标在maker上按下并放开时触发此事件。
mousemove	当鼠标移过maker时触发此事件。
mouseout	当鼠标移出maker时触发此事件。
mouseover	当鼠标移进maker时触发此事件。
rightclick	当鼠标右键单击maker时触发此事件。
touchstart	当在触摸屏上对marker开始进行触摸时触发此事件。
touchmove	当在触摸屏上对marker进行触摸并移动时触发此事件。
touchend	当在触摸屏上对marker触摸完成时触发此事件。

  示例:

  {

  'click': fun,

  'mouseup':fun

  }

- return promise # 通过then进行之后逻辑 返回当前marker对象

```
this.handleMap.setMultiMarker(
    'cameraMark',
    [{
        name:'这是一个监控标点',
        longitude:113.1,
```

```
        latitude:23.1,
        wd: 30,
        hg: 30,
        src: 'imgPath'
    }],
    {
        'click':function () {
            // this 为当前这个标点
            // this.data 为当前标点的数据
        }
    }
).then(res=>{console.log(res)})
```

设置单个mark的样式 setOneMarkerStyle(that *必填, style *必填)

that marker对象

style:{

wd: 30,
hg: 30,
src: 'imgPath'

}

开启指定Mark的拖拽 oepnMarkDrop(that *必填)

that marker对象

关闭拖拽 closeMarkDrop(that *必填)

that marker对象

获取拖拽后的经纬度 getAfterDragging(that *必填)

that marker对象

设置多个mark落点 setMultiMarker (coverageName,data,addEventlist)

落点之前会清除这个图层之前的点

- coverageName *必填 # string 图层名字
- data *必填 # Array 落点的数据

  为数组包对象格式，每个对象中要包含longitude和latitude 作为落点依据

  数据格式：

  [{

  ....

  longitude:113.1,

  latitude:23.1,

  // imgInfo 落图标记设置

  wd: 30 宽度 type Number

  hg: 高度 type Number

  src: 'imgPath' 图片路径 type String

label:'标注',

color:'' 颜色16进制

fontSize:'' 字体大小

textPosition:'' 显示位置 'tp' 顶部 'bt' 底部

}]

- addEventlist # Object 给每个落点绑定事件的对象

  支持事件:

事件名	触发场景
click	当鼠标单击maker时触发此事件。
dblclick	当鼠标双击maker时触发此事件。
mousedown	当鼠标在maker上按下时触发此事件。
mouseup	当鼠标在maker上按下并放开时触发此事件。
mousemove	当鼠标移过maker时触发此事件。
mouseout	当鼠标移出maker时触发此事件。
mouseover	当鼠标移进maker时触发此事件。
rightclick	当鼠标右键单击maker时触发此事件。
touchstart	当在触摸屏上对marker开始进行触摸时触发此事件。
touchmove	当在触摸屏上对marker进行触摸并移动时触发此事件。
touchend	当在触摸屏上对marker触摸完成时触发此事件。

  示例:

  {

  'click': fun,

  'mouseup':fun

  }

- return promise # 通过then进行之后逻辑 返回对应的marker对象

```
this.handleMap.setMultiMarker(
    'cameraMark',
    [{
        name:'这是一个监控标点',
        longitude:113.1,
        latitude:23.1,
        wd: 30,
        hg: 30,
        src: 'imgPath'
    }],
    {
        'click':function () {
```

```
            // this 为当前这个标点
            // this.data 为当前标点的数据
        }
    }
).then(res=>{console.log(res)})
```

指定某个图层添加点 assignCoverageAddMark(coverageName,data,addEventlist='')

- coverageName *必填 # string 图层名字
- data *必填 # Object 落点的数据

  longitude和latitude 作为落点依据，数据格式：

  {

  longitude:113.1,

  latitude:23.1,

  // imgInfo 落图标记设置

  wd: 30 宽度 type Number

  hg: 30 高度 type Number

  src: 'imgPath' 图片路径 type String

  }

- addEventlist # Object 给每个落点绑定事件的对象

  支持事件：

事件名	触发场景
click	当鼠标单击maker时触发此事件。
dblclick	当鼠标双击maker时触发此事件。
mousedown	当鼠标在maker上按下时触发此事件。
mouseup	当鼠标在maker上按下并放开时触发此事件。
mousemove	当鼠标移过maker时触发此事件。
mouseout	当鼠标移出maker时触发此事件。
mouseover	当鼠标移进maker时触发此事件。
rightclick	当鼠标右键单击maker时触发此事件。
touchstart	当在触摸屏上对marker开始进行触摸时触发此事件。
touchmove	当在触摸屏上对marker进行触摸并移动时触发此事件。
touchend	当在触摸屏上对marker触摸完成时触发此事件。

  示例：

  {

  'click': fun,

  'mouseup':fun

}

- return promise # 通过then进行之后逻辑

```
this.handleMap.setMultiMarker(
    'cameraMark',
    [{
        name:'这是一个监控标点',
        longitude:113.1,
        latitude:23.1
    }],
    {
        'click':function () {
            // this 为当前这个标点
            // this.data 为当前标点的数据
        }
    },
    {
        wd: 30,
        hg: 30,
        src: 'imgPath'
    }
).then(res=>{console.log(res)})
```

删除某个指定点 removeMark (that)

- that *必填 要删除的数据对象

生成mark标记 creatTempMark (data,addEventlist='')

- data *必填 # Object 落点的数据

  longitude和latitude 作为落点依据，数据格式：

  {

  longitude:113.1,

  latitude:23.1,

  // imgInfo 落图标记设置:

  wd: 30 宽度 type Number

  hg: 30 高度 type Number

  src: 'imgPath' 图片路径 type String

  }

- addEventlist # Object 给每个落点绑定事件的对象

  支持事件：

事件名	触发场景
click	当鼠标单击maker时触发此事件。
dblclick	当鼠标双击maker时触发此事件。
mousedown	当鼠标在maker上按下时触发此事件。
mouseup	当鼠标在maker上按下并放开时触发此事件。
mousemove	当鼠标移过maker时触发此事件。
mouseout	当鼠标移出maker时触发此事件。
mouseover	当鼠标移进maker时触发此事件。
rightclick	当鼠标右键单击maker时触发此事件。
touchstart	当在触摸屏上对marker开始进行触摸时触发此事件。
touchmove	当在触摸屏上对marker进行触摸并移动时触发此事件。
touchend	当在触摸屏上对marker触摸完成时触发此事件。

示例：

```
{
   'click': fun,
   'mouseup':fun
}
```

构建聚合图标并落点 drawConverge(key,data,ImgInfo,addEventlist)

构建这个图层不需要先新建图层，自动会进行创建

参数	类型	子参数	默认值	必填
key	string	名字	”	*
data	array		[]	*
		longitude ({number} 图标的经度)		*
		latitude ({number} 图标的纬度)		*
		label ({String}图标的标注)		
		color ({String} 标签字体颜色，默认值”#fff”。)		
		fontSize ({String} 标签的字体大小。默认值“12px”，最小为12px。)		
		fontWeight ({String} 标签的字体粗细，可设值有：lighter（细字符）、normal（标准字符）、bold（粗体字符）、bolder（更粗的字符）。默认值“normal”。)		
		fontStyle ({String} 标签的字体样式。可设值有：normal（标准的字体）、italic（斜体）、oblique（倾斜体）。默认值“normal”。)		
		labelAlign ({String} 标签对齐，是由两个字符组成的字符串，如：”lt”, “cm”, “rb”。 其中第一个字符代表水平方向上的对齐，”l”=left, “c”=center, “r”=right。 第二个字符代表垂直方向上的对齐，”t”=top, “m”=middle, “b”=bottom。默认是：‘cb’)		
		labelXOffset ({Number} 标签在x轴方向的偏移量，默认值0。)		
		labelYOffset ({Number} 标签在y轴方向的偏移量，默认值-24。)		
ImgInfo	object		{}	*
		img (图片路径)		*
		width (图标宽度)		*
		height (图标高度)		*
addEventlist	object		{}	
		moveend（聚合图层的缩放事件）		
		clusterend（聚合完成之后触发的事件）		
		click（图标的点击事件）		
		clickout（图标的点击结束后事件，只激活一次）		
		over（图标的鼠标移入事件）		

参数	类型	子参数	默认值	必填
		out（图标的鼠标移出事件）		
		dblclick（图标的鼠标双击事件）		

获取触发事件的图标信息，请在书写事件时在方法中加入形参，例如：

```
{
   click:function(f){
      console.log(f)
   }
}
```

聚合数据筛查 dataScreening(key,condition,conditionData)

参数	类型	说明	必填
key	string	构建聚合图形的key	*
condition	string	筛选条件	*
conditionData	*	筛选数据	*

聚合根据数据修改样式 modifyStyle(key,condition,conditionData,ImgInfo)

参数	类型	说明	必填
key	string	构建聚合图形的key	*
condition	string	筛选条件	*
conditionData	*	筛选数据	*
ImgInfo		参照构建聚合图标并落点 drawConverge中的 ImgInfo	*

聚合图层点击时设置样式 clickAggregation (e,t,ImgInfo)

参数	类型	说明	必填
e	object	当前点数据	*
t	object	当前点	*
ImgInfo		参照构建聚合图标并落点 drawConverge中的 ImgInfo	*

清除聚合图层数据 clearDrawConvergeData（key）

plotting

生成vector图层 createdVectorCoverage (coverageName,style)

- coverageName *必填 # type string 图层名字
- style # object 定义图层的初始样式，也可之后再进行更改可使用 setVectorCoverage 进行更改

参数	类型	默认值	必填
strokeColor	string 线条颜色 （16进制）	'#ffffff'	
strokeWidth	number 线条宽度	2	
fillColor	string 图层覆盖物颜色（16进制）	'#87ceeb'	
fillOpacity	number 图层透明度 (0-1)	0.3	
fontColor	string 字体颜色（16进制）	'#1afa29'	
fontSize	string 字体大小	'20px'	

- return promise # 通过then进行之后逻辑

一个vector图层可以存放很多种类型的标绘图形，圆、矩形、多边形、单箭头、双箭头、部署线....等,使用newDrawFeature 方法在这个图层上添加你需要的标绘图形控件

实例化绘制图形控件 newDrawFeature (coverageName,TypeStr,addEventlist='',repeate)

-

参数	类型	说明	必填
coverageName	string 图层名字		*
TypeStr	string 标绘类型		*
	Path	线	
	CircleEx	圆	
	PolygonEx	多边形	
	Rectangle	矩形	
	PolyLineEx	线	
	DoubleArrow	双箭头	
	DiagonalArrow	斜箭头	
addEventlist	Object	要绑定的事件	
repeate	boolean	默认false只能绘制一次,true为多次绘制	

支持事件:

事件名	触发场景
featureadded	当要素绘制成功时触发此事件
beforefeatureadded	当要素绘制之前触发此事件

示例：

```
{
   'featureadded': fun,
   'beforefeatureadded':fun
}
```

- return promise # 通过then进行之后逻辑

激活标绘图层控件 activateDrawFeature (coverageName)

- coverageName *必填 # type string 激活图层名字

只有指定激活的控件可以进行绘制，其他控件为关闭状态

关闭激活图层控件 closeActivateDrawFeature (coverageName)

- coverageName *必填 # type string 关闭激活图层名字

打开指定vector图层进行图层中的图形编辑 openVectorCoverageEdit (coverageName)

- coverageName *必填 # type string 图层名字
- return promise # 通过then进行之后逻辑

关闭指定vector图层进行图层中的图形编辑 closeVectorCoverageEdit (coverageName)

- coverageName *必填 # type string 图层名字
- return promise # 通过then进行之后逻辑

删除选中图形 delectSelectGraph (coverageName)

- coverageName *必填 # type string 图层名字
- return promise # 通过then进行之后逻辑

清空当前图层 clearAtPresentVectorData (coverageName)

- coverageName *必填 # type string 图层名字
- return promise # 通过then进行之后逻辑

设置图形样式 setVectorCoverage (coverageName,style)

参数	类型	是否必填
coverageName	string 图层名	*
style	object 请参照👇	*

参数	类型	默认值	必填
strokeColor	string 线条颜色 （16进制）	'#ffffff'	
strokeWidth	number 线条宽度	2	
fillColor	string 图层覆盖物颜色（16进制）	'#87ceeb'	
fillOpacity	number 图层透明度 (0-1)	0.3	
fontColor	string 字体颜色（16进制）	'#1afa29'	
fontSize	string 字体大小	'20px'	

绑定监听事件 drawFeaturebindingMonitorEvent (coverageName,addEventlist=",repeate)

只用于绘制控件或绘制图形绑定事件

参数	类型	说明	是否必填
coverageName	string 图层名		*
addEventlist	object 绑定的事件		
	featureadded	当要素绘制成功时触发此事件	
	beforefeatureadded	当要素绘制之前触发此事件	
repeate	boolean		

渲染图形 renderGraph (coverageName,data,label,addEventList)

参数	类型	默认值	是否必填
coverageName	string 图层名		*
data	Array 数据		*
label	string 用哪个字段作为显示该图形的名字	textLabel	
addEventList	object 绑定的事件		

数据示例:

```
data:[
  {
    type: "SuperMap.Geometry.GeoCircle",
    cps: {
      controlPoints: [
        { x: 107.91209220888, y: 27.808370590211 },
          { x: 112.26267814638, y: 24.116964340211 }
      ]
```

```
    },
      textLabel:'测试',
      style:{
        strokeColor: 'skyblue', //生成随机颜色
       strokeWidth: 2,
       fillColor: 'skyblue',
        fillOpacity:0.5,
        fontColor: 'pink',
        fontSize: "20px",
      }
    }
  ]
addEventList:{
  click:(info,f){},
  clickout:(info,f){},
  over:(info,f){},
  out:(info,f){},
  dblclick:(info,f){},
}
```

这里定义style将会成为这个图形的最终样式，如果没有定义将使用随机颜色和默认数据，对于style参数请参考：“设置图形样式 setVectorCoverage” 中的style参数

type:

GeoCircle 圆　GeoPolygonEx 多边形　GeoRectangle 矩形　GeoPolyline 线
GeoDoubleArrow 双箭头　GeoDiagonalArrow 斜箭头

渲染多边形 renderPolygon (coverageName,data,addOrRemoveFlag)

参数	类型	默认值	是否必填
coverageName	string 图层名		*
data	Array 数据		*
addOrRemoveFlag	boolean 如果这个图层已有数据可选择清除	false	

data数据类型如下:

```
[
  [[119.49359134318365, 32.73196912621176],[119.49359134318365, 32.73196912621176]],
  [[119.49359134318365, 32.73196912621176],[119.49359134318365, 32.73196912621176]]
]
```

绘制闭合多边形时想中间闭合请传入2组数据，一组为外边数据，另外一组为内边，绘制出来时中间是没有矢量覆盖物遮罩。

渲染缓冲区renderBuffer(coverageName,data,value,addOrRemoveFlag)

参数	类型	默认值	是否必填
coverageName	string 图层名		*
data	Array 数据		*
value	number 数字	20 单位: m	
addOrRemoveFlag	boolean	false	

data 数据获得：使用 getCoverageData 获得当前图层数据 res[num] 某个图形下的的数据，一般是 res.length-1 的数据 res[num].geometry.components

addOrRemoveFlag 绘制下个缓冲区时删除上个缓冲区

return 缓冲区外包边图形的每个点数据，不闭合图形为单个数据，如果是闭合图形一般为两个数组，具体数据请使用获得，不做呈现。

通过一个经纬度和距离，获取圆圈另外的点 getCirLonAndLat (data)

参数	类型	是否必填
data	Object	*
	longitude 经度	*
	latitude 纬度	*
	r 半径或距离 单位M	*

res:{

longitude,

latitude,

rDeg // 把r 转换为 经纬度 的数据，可以直接用在经纬度的加减上面

}

绘制圆圈 vectorDrawCir (coverageName,data,style)

参数	类型	默认值	是否必填
coverageName	string 绘制的图层名		*
data	Object 绘制圆需要的数据		*
	longitude 经度		*
	latitude 纬度		*
	r 半径或距离单位M		*
style	Object 绘制圆圈的样式		
	strokeColor 线的颜色		
	strokeWidth 线的宽度		
	fillColor 区域颜色		
	fillOpacity 区域的透明度		

绘制线 vectorDrawLine (coverageName,stated,end,style)

参数	类型	默认值	是否必填
coverageName	string 绘制的图层名		*
stated	object 参照calculateTowDistance 的stated		*
end	object 参照calculateTowDistance 的end		*
style	object 线的样式		
	strokeColor String	"#fff"	
	strokeWidth Number	1	
	strokeDashstyle String 可选值有：dot,dash,dashot,longdash,longdashdot,solid	"solid"	

批量绘制线 batchDrawVectorLine（coverageName,data）

参数	类型	默认值	是否必填	
coverageName	string 绘制的图层名		*	
data	Array 绘制线的数据			
	stated object 参照calculateTowDistance 的 stated			
	end object 参照vectorDrawLine 的 end			
	style object 线的样式参照vectorDrawLine 的 style			

示例：

```
let data = [
 {
  stated: {...},
  end: {...},
  style:{...}
 }
];
```

绘制可编辑的线和面图形 drawLinePolyg (coverageName，addEventlist)

参数	类型	说明	是否必填
coverageName	String	图层名	*
addEventlist	object	对象	
	Line、Polygon	线和面对象支持的事件	
	beforefeatureadded	绘制前触发	
	featureadded	绘制完成触发	
	Modify	进行编辑线或面图形时触发	
	beforefeaturemodified	当图层上的要素开始编辑前触发该事件。	
	featuremodified	当图层上的要素编辑时触发该事件。	
	afterfeaturemodified	当图层上的要素编辑完成时，触发该事件。	

addEventlist的结构示例

feature 为图形对象，地图Api会传参过来

```
addEventlist:{
```

```
    Line:{
        beforefeatureadded:eventFun （feature） {}
        featureadded:eventFun（feature） {}
    },
    Polygon:{}
    Modify:{}
}
```

对地图操作前请先生成图层createdVectorCoverage，如已生成图层可以忽略
绘制可编辑的线和面图层，只能用来绘制线和面，如果当前图层已经存在除线和面以外的图形，请重新建立一个新图层，以免出错

1.开启线、面的绘制 openLinePolygon(coverageName,type,falg)

参数	类型	说明	是否必填
coverageName	String	图层名	*
type	String	类型	*
	line	线	
	polygon	面	
falg	boolean	开启或者关闭	*

2. 控制线和面的编辑开启和关闭 activeACloseLinePolyg(coverageName,flag)

参数	类型	说明	是否必填
coverageName	String	图层名	*
flag	boolean	开启或者关闭	*

3.删除控制线和面的constrol deleteConstrol(coverageName)

参数	类型	说明	是否必填
coverageName	String	图层名	*

根据条件 获取指定的数据 getSpecifiedData (coverageName, geometryId)

参数	类型	是否必填
coverageName	String 图层名	*
geometryId	String 图形id	*

根据条件删除某个指定的数据 deleteSpecifiedData(coverageName, geometryId)

参数	类型	是否必填
coverageName	String 图层名	*
geometryId	String 图形id	*

给指定vector图层 添加单个数据 specifyVectorLayerAdd(coverageName, features)

参数	类型	是否必填
coverageName	String 图层名	*
features	Object 图形对象	*

设置单个vector样式 setSingleStyle(features, st, editFlag)

参数	类型	是否必填
coverageName	String 图层名	*
features	Object 图形对象	*
editFlag	Boolean控制读取的id，图形是可编辑的线面时为true，其他图形都为false，默认为false	

设置单个点setPoint(coverageName, data)

参数	类型	是否必填
coverageName	String 图层名	*
data	Object 图形对象	*
	longitude	
	latitude	
	style	

初始化生成热力图图层initHeatMapLayer(coverageName)

参数	类型	是否必填
coverageName	String 图层名	*

生成热力图上的热点createHeatPoints(coverageName, data, radius)

参数	类型	是否必填
coverageName	String 图层名	*
data	Array 落点数据	*
	每一条数据必须包含longitude and latitude、value	
radius	半径大小px 默认是1	

清除热力图数据clearHeatPoints(coverageName)

参数	类型	是否必填
coverageName	String 图层名	*

图层公共

获取地图最大的缩放层级 getMapMaxZoom

控制整个图层的文字显示 setCoverTextVis（flag)

参数	类型	是否必填
flag	Boolean 控制显示隐藏	*

控制整个图层的文字颜色 setCoverTextColor（color)

参数	类型	是否必填
color	string 文字颜色	*

控制所有图层的显示隐藏 setAllCoverageShowHide（flag)

参数	类型	是否必填
flag	Boolean 控制显示隐藏	*

控制图层的显示 setCoverageShow (coverageName)

参数	类型	是否必填
coverageName	String 图层名	*

控制图层的隐藏 setCoverageHide (coverageName)

参数	类型	是否必填
coverageName	String 图层名	*

批量控制图层的显示 setBatchCoverageShow (coverageNameArr)

参数	类型	是否必填
coverageNameArr	Array 图层名	*

示例：coverageNameArr:['coverageName1','coverageName2','coverageName3']

批量控制图层的隐藏 setBatchCoverageHide (coverageNameArr)

参数	类型	是否必填
coverageNameArr	Array 图层名	*

示例：coverageNameArr:['coverageName1','coverageName2','coverageName3']

计算两点之间的距离 calculateLineDistance(start, end)

批量计算两点之间的距离batchCalculateTowDistance （data）

参数	类型	是否必填
data	Array	*

示例：

```
let data = [
  {
    stated: {
      longitude: 113.8961568826,
      latitude: 22.59911272774
    },
    end: {
      longitude: 113.29709371999928,
      latitude: 23.68697097127243
    }
  },
  {
    stated: {
      longitude: 113.8961568826,
      latitude: 22.99911272774
    },
    end: {
      longitude: 113.29709371999928,
      latitude: 23.98697097127243
    }
  }
];
batchCalculateTowDistance(data)
```

return [Number,Number]

获取当前图层的数据 getCoverageData (coverageName)

参数	类型	是否必填
coverageName	String 图层名	*

return 一个Array的数据，包含了当前图层标绘图形的数据

清除并删除图层 clearDeleteCoverage(coverageName)

参数	类型	是否必填
coverageName	String 图层名	*

切换图层 tabCoverage（name）

参数	类型	是否必填
name	String 图层名 （影像、 矢量）	*

圈选 handleCheckAll （TypeStr）

参数	类型	说明	必填
TypeStr	string 标绘类型		*
	CircleEx	圆	
	PolygonEx	多边形	
	Rectangle	矩形	

开启点击获取落点信息 clickDroppoint (cb) 逆地理

cb作为回调函数，因为获取点信息要和服务端交互获取数据，还有一些其他逻辑所以这里不使用promise，使用callback的方式进行获取数据和后续的逻辑，res就是从服务器中拿到的数据，示例：

```
clickDroppoint(function(res){
    console.log(res)
})
```

关闭点击获取点信息closeClickDroppoint() 通过then知道调用完成

还原默认鼠标样式 restoreDefaultStyle

恢复成初始状态（手）

设置鼠标样式 setMouseStyle (src)

src：图片路径

设置地图中心点 setCentent （data,zoom）

data *必填 # Object 落点的数据

zoom *必填 # Number 显示层级

longitude和latitude 作为落点依据，数据格式：

{

longitude:113.1,

latitude:23.1

}

添加自定义图片到图层 addImgCoverage(url, bounds,key='')

参数	类型	说明	是否必填
url	string	'xxx/xxxx.png'	*
bounds	string	示例: '-180,-90,180,90' 左下角和右上角的经纬度 两个点的经纬度不能搞错，否则没有图像出现也没有报错 !!!	*
key	string	这个图层的名字	

在地图上添加标注文字 addFontToCoverage(obj)

obj 是一个对象类型的数据主要的键为：

参数	类型	说明	是否必填
lon	Number	经度	*
lat	Number	纬度	*
key	string	标注的文字	*
color	string	颜色，支持16进制、rgb、rgba	
size	string	大小 '100px'	
bgr	string	颜色，支持16进制、rgb、rgba	
opacity	Number	0-1	

搜索POI数据 searchPOIdata(key)

参数	类型	说明	是否必填
key	string		*

截图 mapScreenshot()

地图上的图片如果有http或者https的资源大概率不能被canvas绘制下来，请使用本地资源

计算面积 calculateAreas(geometry)

geometry 为feature当前图形的数据，请参照 drawLinePolyg 方法

路劲规划 pathPlanning(obj,type)

参数	类型	说明	是否必填
obj	Object 数据对象		
type	String	规划类型	*
	'drive'	驾车规划	
	'transit'	公交规划	
	'cycling'	骑行规划	
	'walk'	步行规划	
	'truck'	货车规划	
objData	根据类型所需要的数据👇	根据类型所需要的数据👇	
	origin String	起点 所有类型都需要	*
	destination String	终点 所有类型都需要	*
	waypoints String	途经点 驾车规划 选填	
	city String '北京'	城市名 公交规划 必填	
	size Number	车辆类型 货车规划 选填1：微型车，2：轻型车（默认值），3：中型车，4：重型车	

popup

创建打开弹窗 mapPopup (data)

参数	类型	说明		是否必填
data	Object 数据对象			*
	longitude	Number 经度		*
	latitude	Number 纬度		*
	class	string or HTMLElement 用于展现的模板的类名 例如：'.popup'、'#popup'		*
	id	string 弹窗的唯一标识		*

关闭、清除弹窗 closeClearPopup(id)

坐标转换

高德转天地图 gdTrTdt(lng,lat)

百度转天地图 bdTrTdt(lng,lat)

天地图转百度 tdtTrBd(lng,lat)

天地图转高德 tdtTrGd(lng,lat)

参数	类型	说明	是否必填
lng	number	经度	*
lat	number	纬度	*

返回的数据:

{

latitude: 23.103166700476518,

longitude: 113.1120644952671

}

地图图层定义和使用

`this.handleMap` 为整个整个项目创建地图时返回的操作函数对象
若地图API文档中没有你需要的函数方法，请根据业务需求，考虑是否增加到地图API中，写入时请按照规范进行，以便阅读和使用！