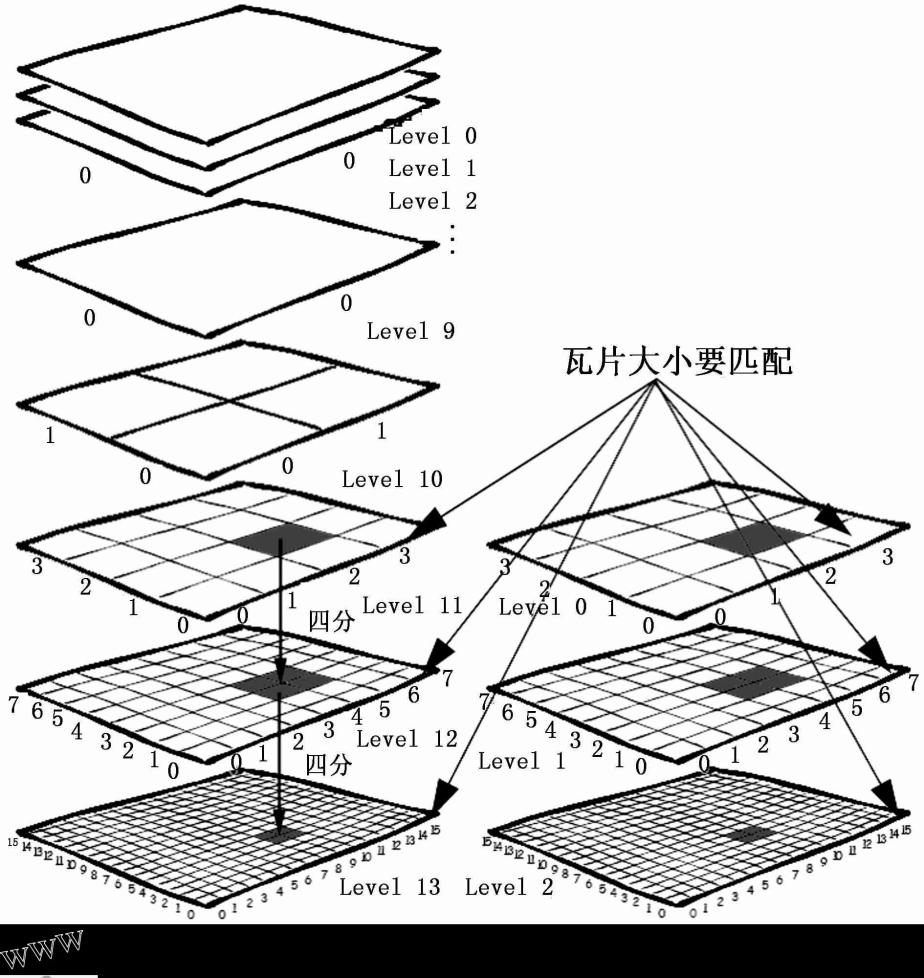
# Openlayers 4加载地图

## Openlayers 4 加载瓦片地图

### 瓦片地图解析

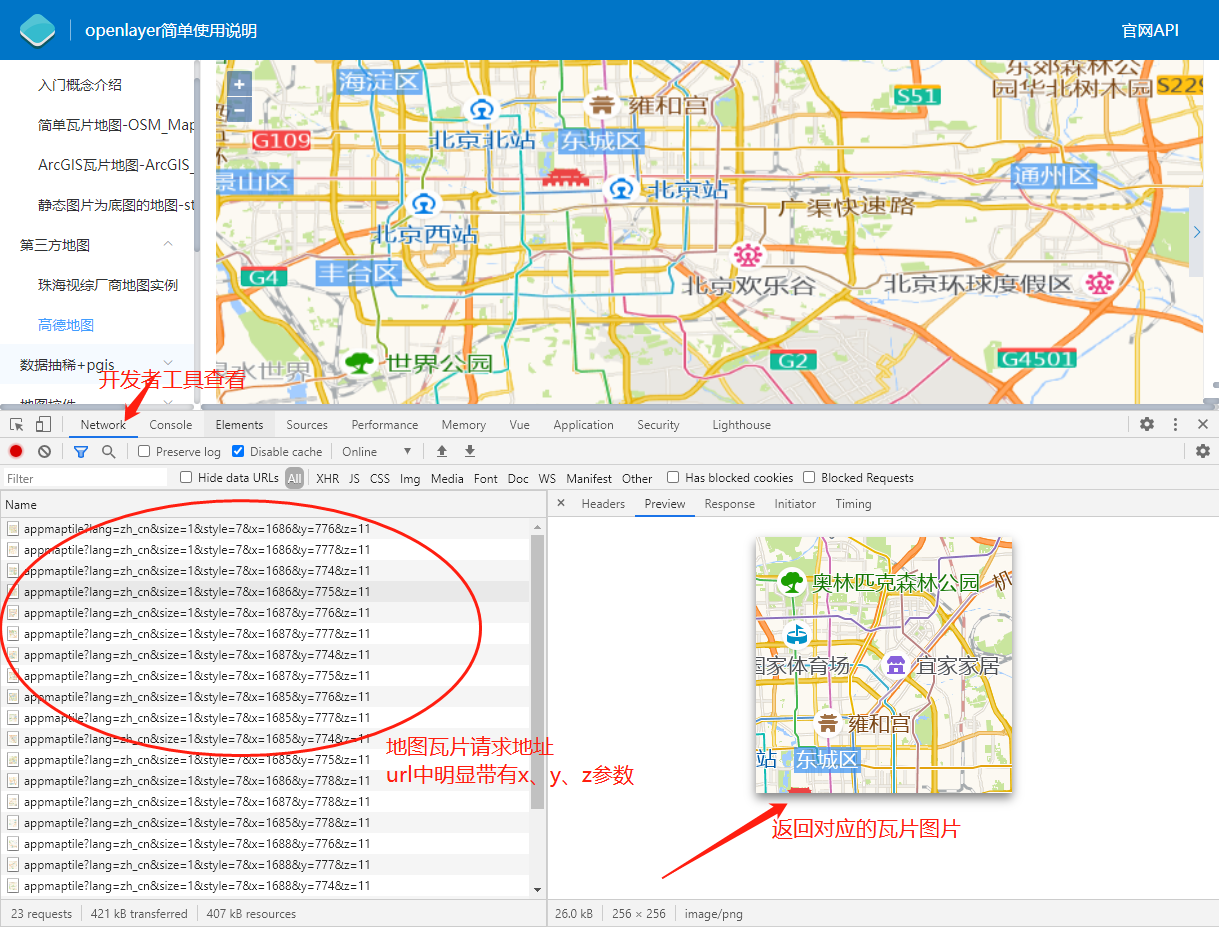
常用的瓦片地图是一个三维的概念，我们通常使用xyz这样的坐标来精确定位一张瓦片。通常z用于表示地图层级，而xy表示某个层级内的瓦片平面，x为横纵坐标，y为纵轴坐标。



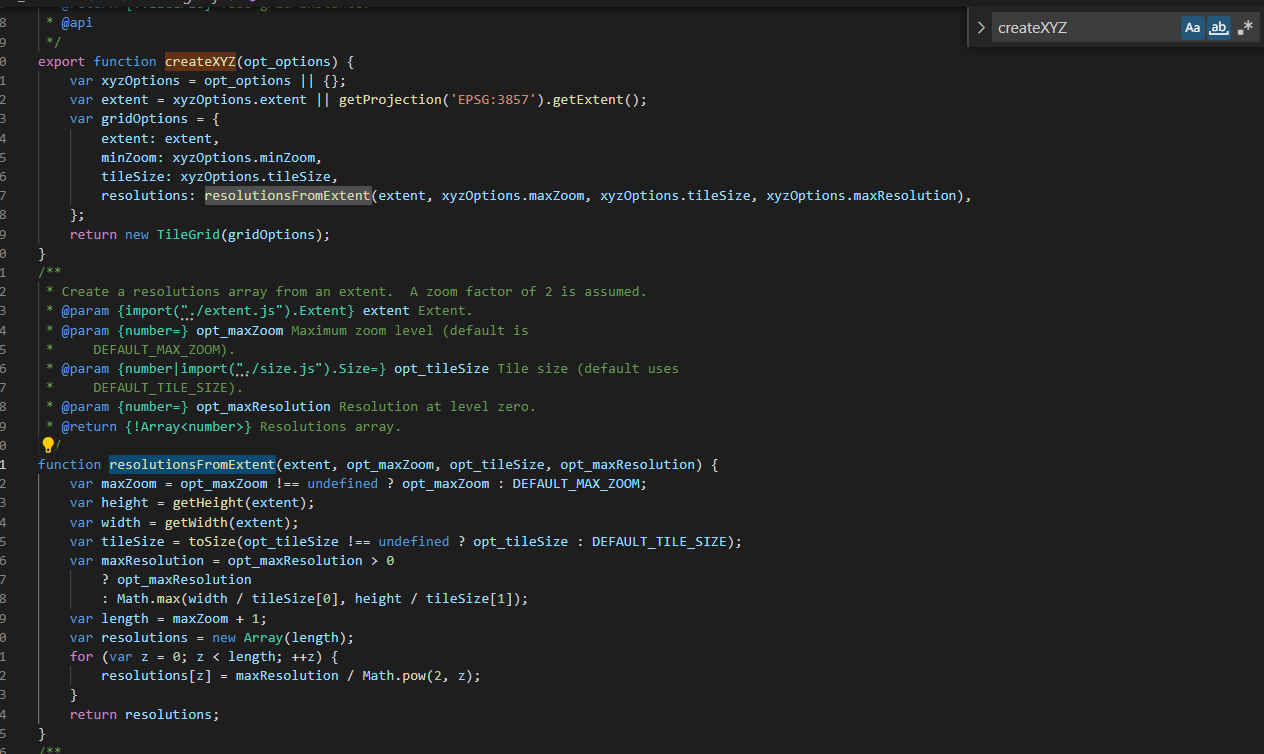
### 瓦片地图辨别

在浏览器中打开任意一个在线的网页地图，然后打开浏览器的开发者工具，再随意拖动，放大缩小地图。之后在开发者工具里查看新发起的请求，是否有一些图片请求，查看请求返回的图片，是否为正在浏览的地图的一部分，如果是，则基本为瓦片地图。

下面以高德地图为例，说明一下在线瓦片地图请求信息：



### 根据信息加载瓦片地图



#### 正常情况下

若地图没有经过特别处理，在已知url的情况下，只需要直接使用xyz进行地图初始化，验证瓦片是否可以正常加载，参考如下：

只需要把url信息对应替换即可。

|  |
| --- |
| <template>    <div id="map" **class**="map" style="height:100%;width:100%;"></div>  </template>  <script>  **import** "ol/ol.css";  **import** Map from "ol/Map";  **import** XYZ from "ol/source/XYZ";  **import** TileLayer from "ol/layer/Tile";  **import** View from "ol/View";  **export** **default** {    name: "GaodeMap",    mounted() {  **const** mapLayer = **new** TileLayer({        source: **new** XYZ({          url: "http://wprd0{1-4}.is.autonavi.com/appmaptile?lang=zh\_cn&size=1&style=7&x={x}&y={y}&z={z}",        }),      })  **new** Map({        target: "map",        layers: [mapLayer],        view: **new** View({          center: [106.51, 29.55],          zoom: 12,          projection: "EPSG:4326",        }),      });    },  };  </script> |

#### 其他情况下

##### 情况1：

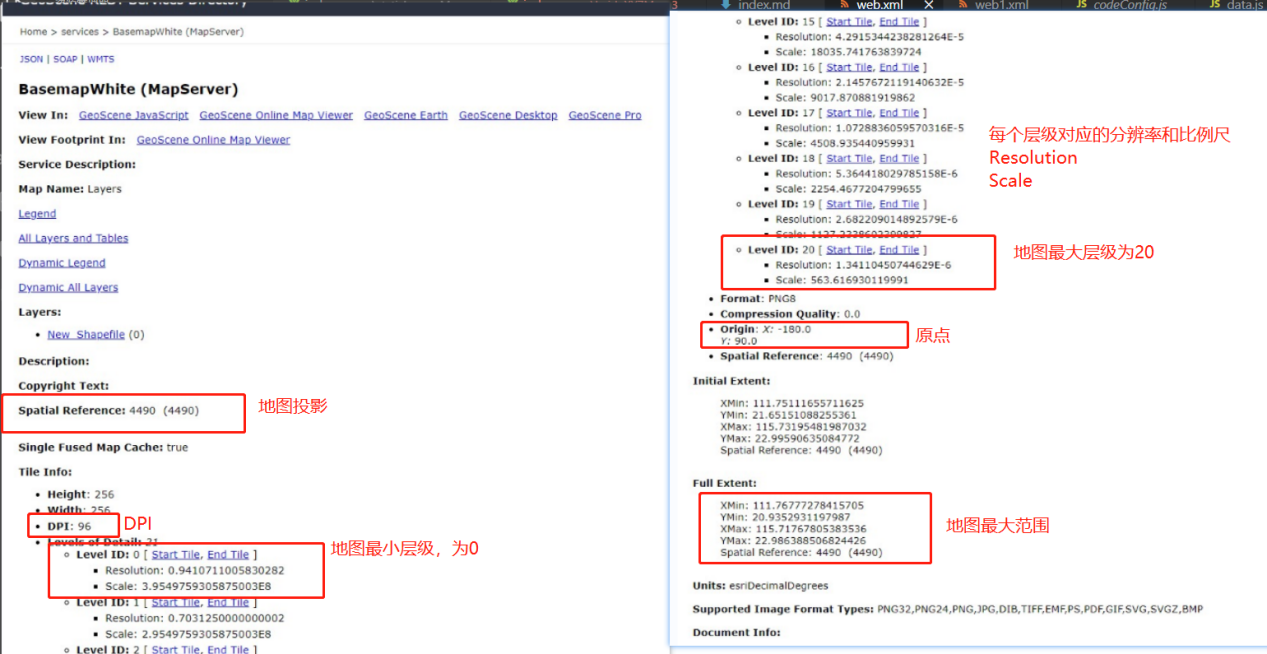
理想情况下，直接客户直接给对出对应的url地址，并提供表格下面对应的信息，就可以加载瓦片地图，\*表示必须提供，若没有提供，则可以尝试使用参考值去测试。

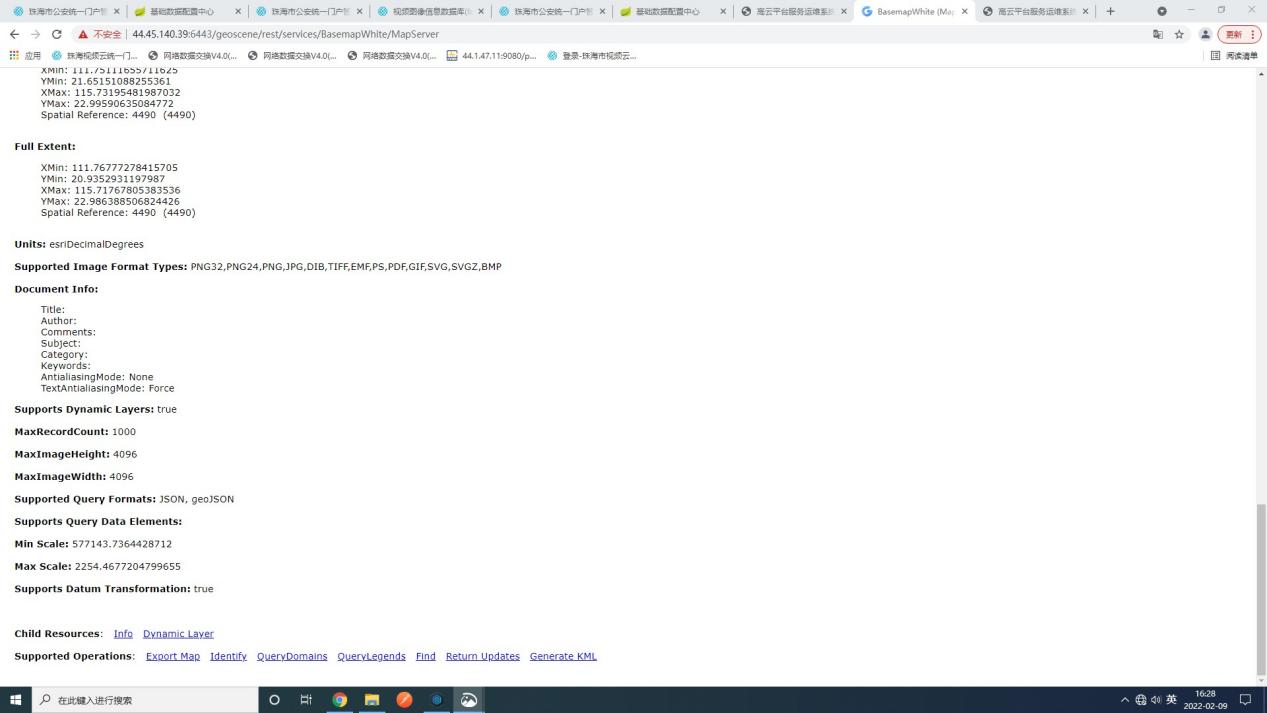
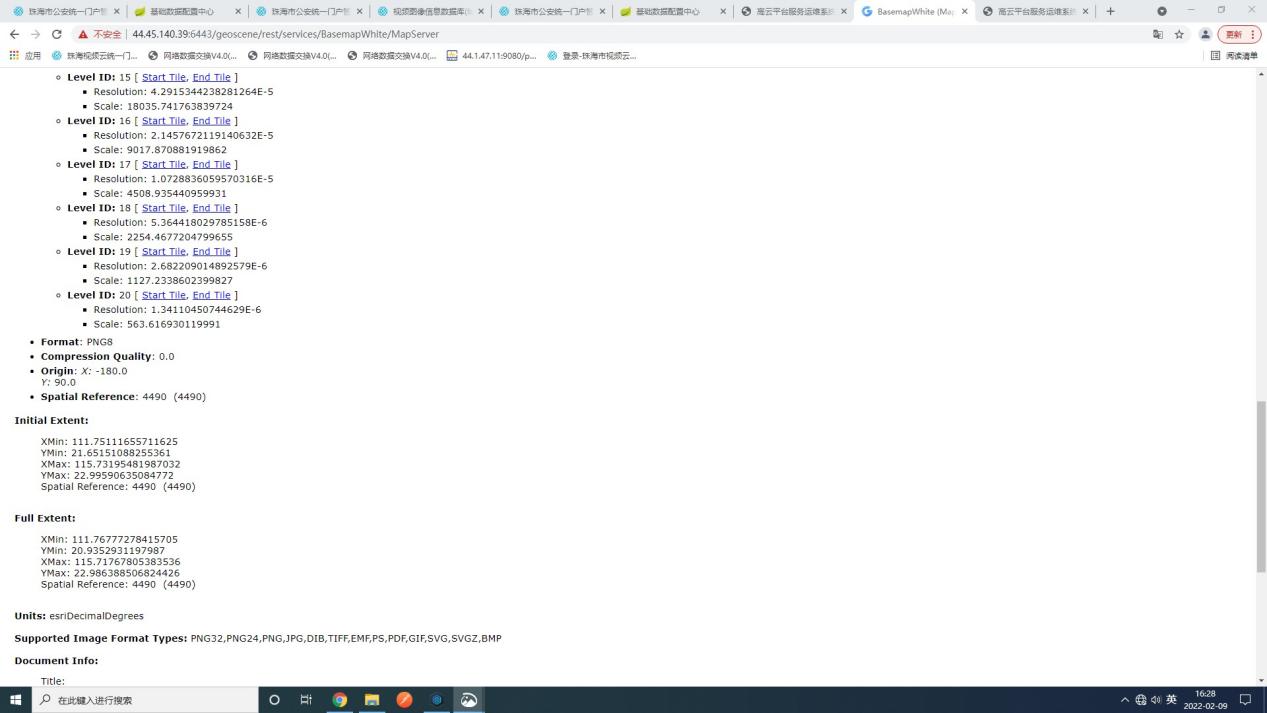
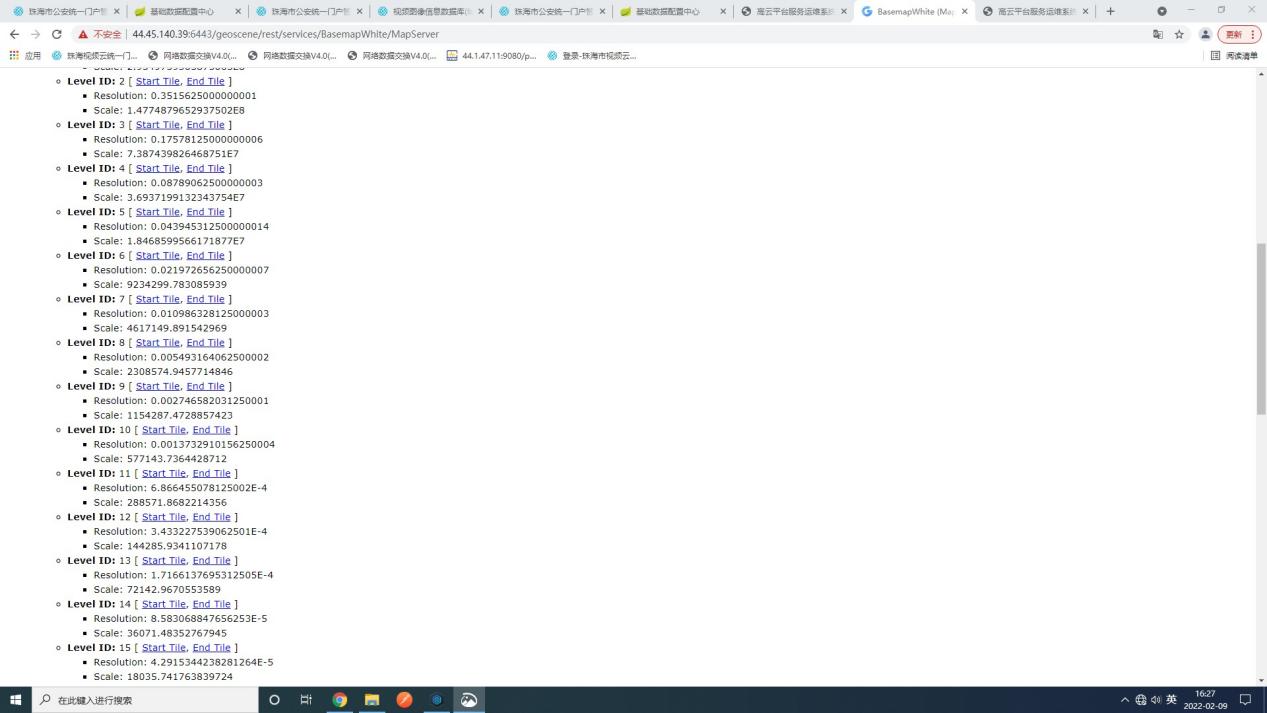
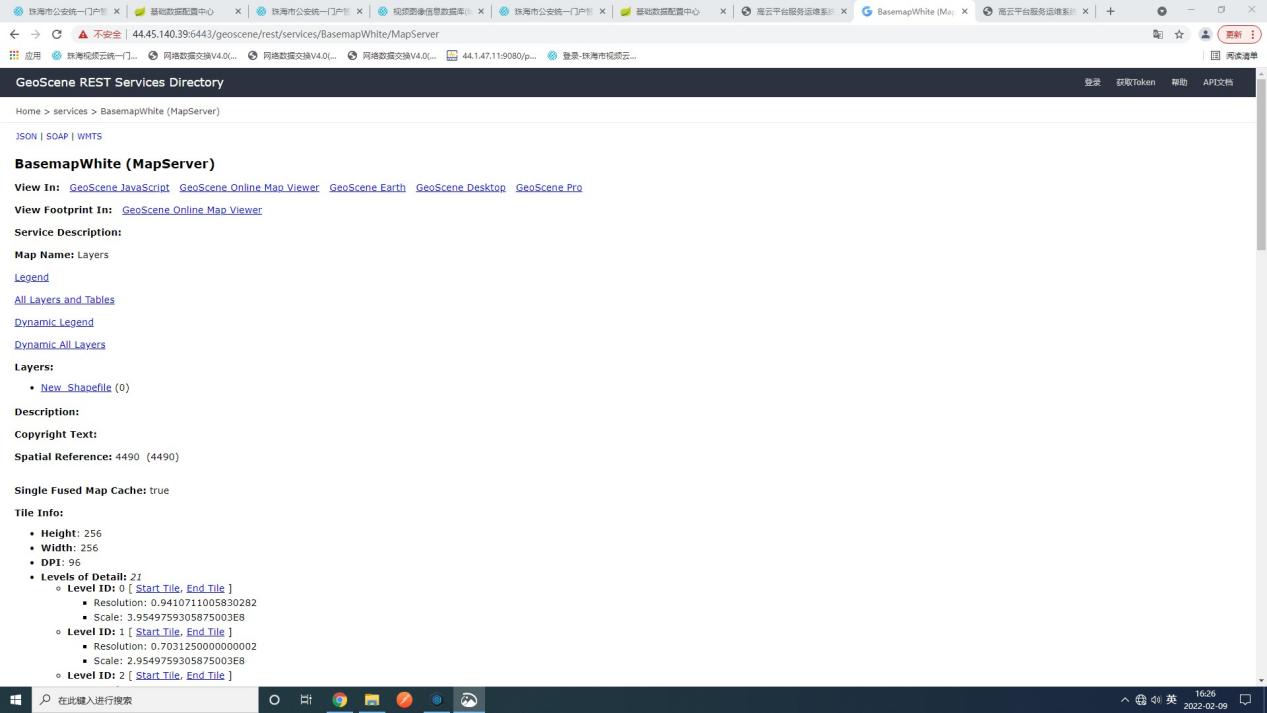
|  |  |
| --- | --- |
| 参数 | 参考值 |
| **坐标原点\*** |  |
| **地图url\*** |  |
| **分辨率\*** |  |
| 投影 | 3857 |
| 切片尺寸 | 256 |
| 地图范围extent |  |

##### **情况2：**

若现场人员不熟悉地图的参数信息，可让现场直接提供wmts.xml文件。下图所示为某地图厂商的wmts文件截图，url需要另外提供，在以下截图中，暂时无法获取url。

从截图中，可以得出以下信息：





##### 情况3：

若现场人员对地图信息完全不了解，或者厂商提供的demo运行不起来，则需要另外想办法

* 建议1：打开已有的厂商地图，f12查看地图加载地址，如果恰巧是xyz的请求格式，那么可以尝试直接用xyz方式加载该url。其中url里面的xyz是变量，改成x={x}模式即可，例如：<http://[IP]:[PORT]/xxxxx?x={x}&y={y}&z={z}>
* 建议2：条件允许的话，让负责人联系第三方厂商，获取url信息

##### 其他情况下地图加载方法

获取到地图瓦片信息以后，即可参考以下方式加载地图：

|  |
| --- |
| <template>    <div id="map" **class**="map" style="height:100%;width:100%;"></div>  </template>  <script>  **import** "ol/ol.css";  **import** Map from "ol/Map";  **import** XYZ from "ol/source/XYZ";  **import** TileLayer from "ol/layer/Tile";  **import** View from "ol/View";  **import** TileGrid from "ol/tilegrid/TileGrid"  **export** **default** {    name: "otherMap",    mounted() {      let mapLayer = **new** TileLayer({          source: **new** XYZ({            tileGrid: **new** TileGrid({              resolutions: [0.9410711005830282,  0.7031250000000002, 0.3515625000000001, 0.17578125000000006, 0.08789062500000003, 0.043945312500000014, 0.021972656250000007,0.010986328125000003, 0.005493164062500002, 0.002746582031250001, 0.0013732910156250004, 6.866455078125002E-4, 3.433227539062501E-4, 1.7166137695312505E-4, 8.583068847656253E-5,4.2915344238281264E-5, 2.1457672119140632E-5, 1.0728836059570316E-5, 5.364418029785158E-6, 2.682209014892579E-6, 1.34110450744629E-6],              origin: [-180, 90],            }),            maxZoom: 20, // 图层最大缩放层级            projection: "EPSG:4490", // 图层对应坐标系对象            tileSize: [256, 256], // 图层对应切片尺寸对象            url: '地图url地址' // 图层对应地图服务url          })        })  **new** Map({        target: "map",        layers: [mapLayer],        view: **new** View({          center: ['地图初始化中心点位'],          zoom: 12,          projection: "EPSG:4326",        }),      });    },  };  </script> |

### 瓦片地图加载之PGIS矢量地图

加载PGIS瓦片地图，也是和上面相似的渲染方法。

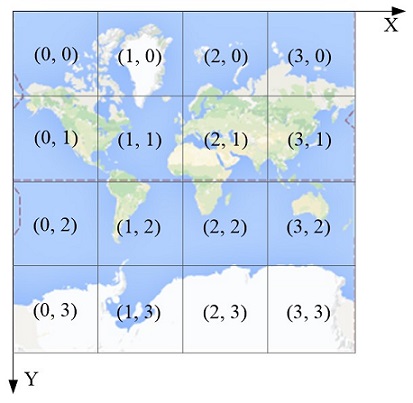


|  |
| --- |
| let pgisSource = **new** TileImage({    projection: 'EPSG:3857',    tileGrid:  **new** TileGrid({      origin: [0,0],    // 设置原点坐标      resolutions: [1,0.5,0.25,0.125,0.0625,0.03125,0.015625,0.0078125,0.00390625,0.001953125,9.765625E-4,4.8828125E-4,2.441406124E-4,1.220703125E-4,6.103515625E-5,3.0517578125E-5,1.52587890625E-5,7.62939453125E-6,3.814697265625E-6,1.9073486328125E-6],  // 设置分辨率      extent: [-180,-90,180,90]    }),  });  let mapLayer = **new** TileLayer({      source: pgisSource  }); |

### 常见瓦片地图加载demo

OpenLayers提供了ol.source.XYZ这种通用的Source来适应广大的在线瓦片地图数据源，具备很好的适用性。通常情况下，开发者想要加载不同的在线瓦片地图源，则只需要更改ol.source.XYZ的构造参数中url就可以了。

在OpenLayers 中，默认使用的瓦片地图的坐标系的原点在左上角，向上为y轴正方向，向右为x轴正方向。



#### (1)高德地图加载

其中GaodeMapLayer 是表示高德地图瓦片图层

|  |
| --- |
| <template>    <div id="map" **class**="map" style="height:100%;width:100%;"></div>  </template>  <script>  **import** "ol/ol.css";  **import** Map from "ol/Map";  **import** XYZ from "ol/source/XYZ";  **import** TileLayer from "ol/layer/Tile";  **import** View from "ol/View";  **export** **default** {    name: "GaodeMap",    mounted() {  **const** mapLayer = **new** TileLayer({        source: **new** XYZ({          url: "http://wprd0{1-4}.is.autonavi.com/appmaptile?lang=zh\_cn&size=1&style=7&x={x}&y={y}&z={z}",        }),      })  **new** Map({        target: "map",        layers: [mapLayer],        view: **new** View({          center: [106.51, 29.55],          zoom: 12,          projection: "EPSG:4326",        }),      });    },  };  </script> |

#### (2)Open Street Map地图加载

Open street map地图，可以直接使用new ol.layer.Tile({source: new ol.source.OSM()})作为Layers图层值进行地图初始化。但是同样，经过对open street map进行分析，它也是瓦片地图，因此同样可以使用ol.source.XYZ来加载Open Street Map地图，初始化结果的一样的。

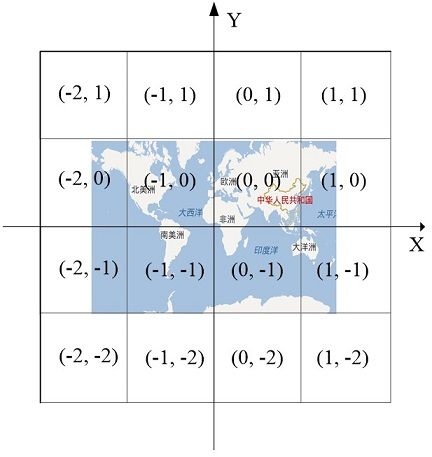
|  |
| --- |
| **const** mapLayer = **new** TileLayer({    source: **new** XYZ({      url: "http://{a-c}.tile.openstreetmap.org/{z}/{x}/{y}.png",    }),  }) |

#### Yahoo地图加载

|  |
| --- |
| **const** mapLayer = **new** TileLayer({    source: **new** XYZ({      url: "https://{0-3}.base.maps.api.here.com/maptile/2.1/maptile/newest/normal.day/{z}/{x}/{y}/512/png8?lg=ENG&ppi=250&token=TrLJuXVK62IQk0vuXFzaig%3D%3D&requestid=yahoo.prod&app\_id=eAdkWGYRoc4RfxVo0Z4B",    }),  }) |

### 瓦片地图加载之百度地图

直接使用ol.source.XYZ中的url值进行初始化地图，这种方法并不使用于所以的在线瓦片地图，例如百度地图。针对原点不在左上角的瓦片地图，那么就需要对瓦片重新定义瓦片坐标系，使用ol.tilegrid.TileGrid类做出一些调整。

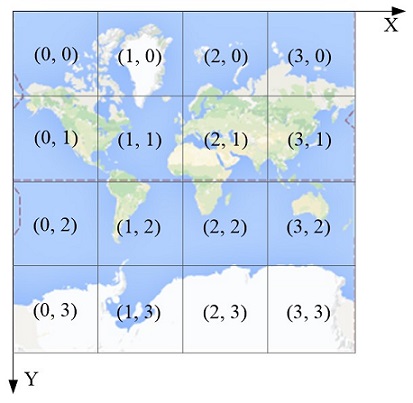


在网上有看到一些例子，百度地图y轴的值的计算是“y=tileCoord[2]”,下面的例子却是“y = -tileCoord[2]-1”，主要是因为以前3、4、5版本加载百度地图的代码用y=tileCoord[2]，openlayers升级到6版本需要用y = -tileCoord[2]-1，至于具体的底层原因，自行查阅。

|  |
| --- |
| <template>    <div id="map" **class**="map" style="height:100%;width:100%;"></div>  </template>  <script>  **import** "ol/ol.css";  **import** Map from "ol/Map";  **import** TileLayer from "ol/layer/Tile";  **import** View from "ol/View";  **import** TileImage from "ol/source/TileImage"  **import** TileGrid from "ol/tilegrid/TileGrid"  **export** **default** {    name: "baiduXYZMap",    mounted() {      // 百度平面坐标系的坐标原点与百度瓦片坐标原点相同，以瓦片等级18级为基准，规定18级时百度平面坐标的一个单位等于屏幕上的一个像素      // 自定义分辨率和瓦片坐标系      let resolutions = [];      let maxZoom = 18;      // 计算百度使用的分辨率  **for**(let i=0; i<=maxZoom; i++){        resolutions[i] = Math.pow(2, maxZoom-i);      }      // 创建百度地图的数据源      let baiduSource = **new** TileImage({        projection: 'EPSG:3857',        tileGrid:  **new** TileGrid({          origin: [0,0],    // 设置原点坐标          resolutions: resolutions  // 设置分辨率        }),        tileUrlFunction: **function**(tileCoord, pixelRatio, proj){          let z = tileCoord[0];          let x = tileCoord[1];          let y = -tileCoord[2]-1;          // 百度瓦片服务url将负数使用M前缀来标识  **if**(x<0){              x = 'M' + (-x);  }  **if**(y<0){              y = 'M' + (-y);  }  **return** "http://online0.map.bdimg.com/onlinelabel/?qt=tile&x="+x+"&y="+y+"&z="+z+"&styles=pl&udt=20160426&scaler=1&p=0";        }      });      // 百度地图层      let mapLayer = **new** TileLayer({          source: baiduSource      });  **new** Map({        target: "map",        layers: [mapLayer],        view: **new** View({          center: [106.51, 29.55],          zoom: 12,          projection: "EPSG:4326",        }),      });    },  };  </script> |

## OSM、谷歌地图各等级下的比例尺和分辨率计算表

经过查证，osm、谷歌地图、高德地图、天地图，都是参照谷歌地图切片方案进行切割，这也是互联网在线地图通用的做法。Google Maps的切片定义中心点在左上角，将地图以x\y\z的方式存储读取，Z表示缩放层级，Z=zoom；XY的原点在左上角，X从左向右，Y从上向下。



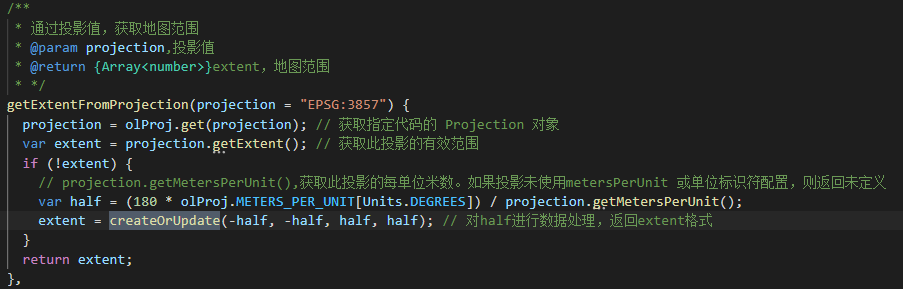
谷歌地图切片方案下的地图属性互相转换

### 1.通过投影，获取地图范围

import \* as olProj from "ol/proj";

import Units from "ol/proj/Units";

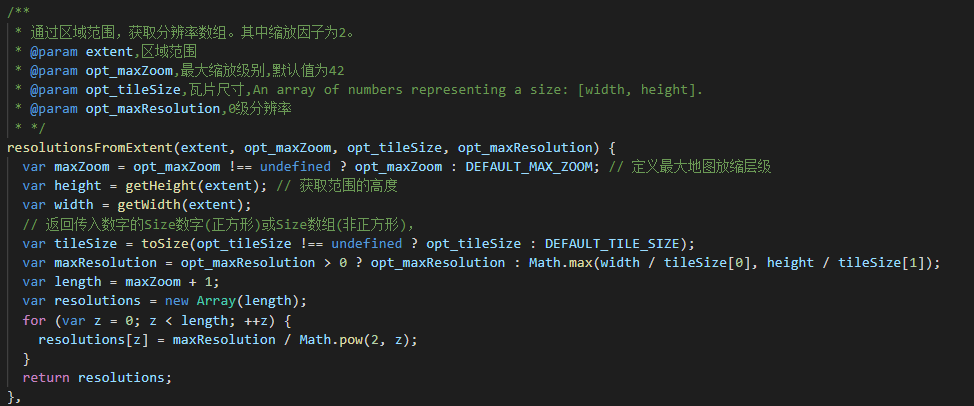
import { createOrUpdate, getHeight, getWidth } from "ol/extent.js";



### 2.通过区域范围，获取分辨率数组。其中缩放因子为2。

var DEFAULT\_MAX\_ZOOM = 42;

var DEFAULT\_TILE\_SIZE = 256;



### 3.通过区域范围，获取地图原点坐标



## openlayers中的投影

openlayer默认是3857投影坐标系

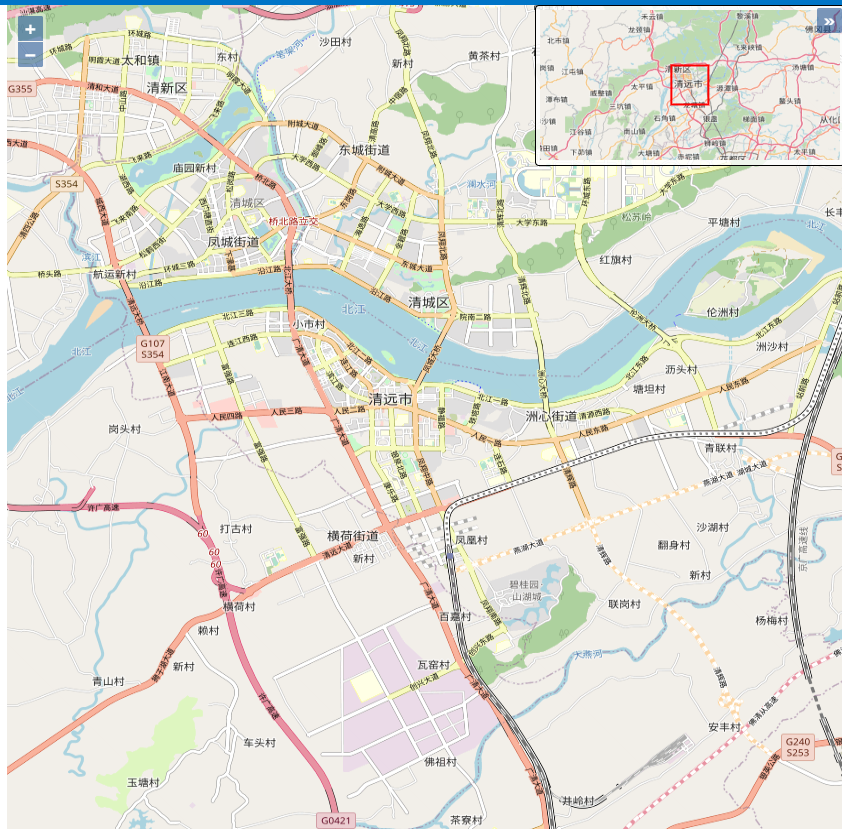
### 1.EPSG:3857和EPSG:4326转换

3857平面坐标转4326经纬度：toLonLat([471983.45, 3490990.75])

4326经纬度转3857平面坐标：fromLonLat([104.704968, 31.540962])

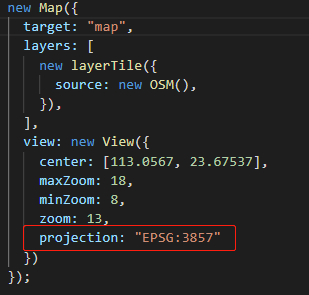
在实际应用中，我们一般都是直接使用4326投影坐标系，所以在初始化地图的时候，不需要对经纬度坐标进行转换。

例如：地图初始化，中心点位为[113.0567, 23.67537]



但是若想要使用3857投影，那么进行地图点位操作的时候，就需要对经纬度坐标转为3857坐标，才可以进行正常的显示

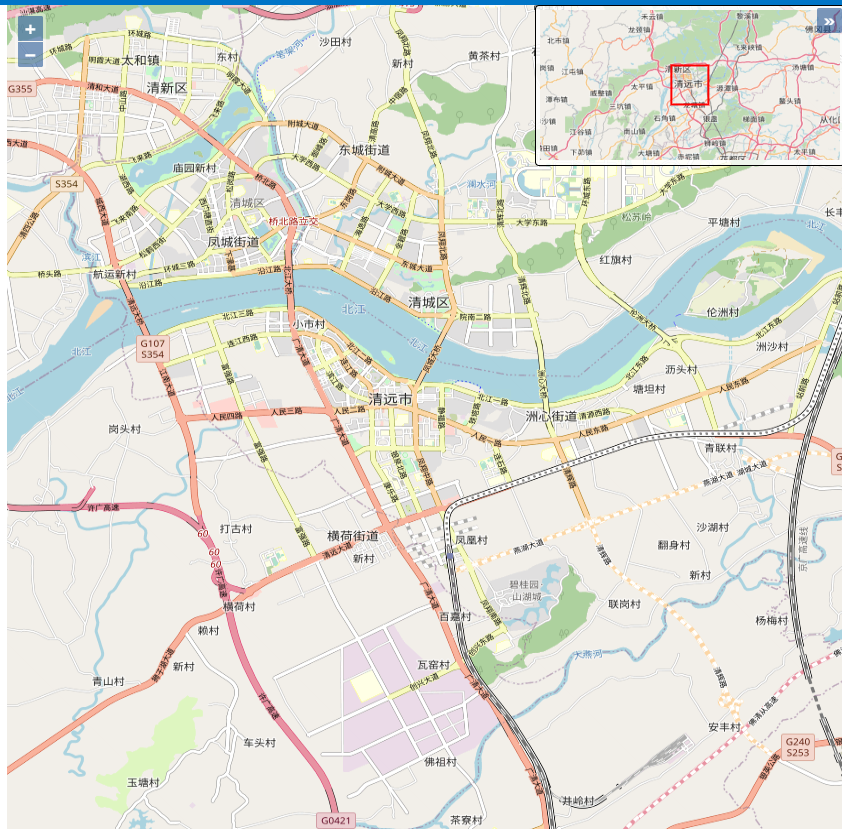
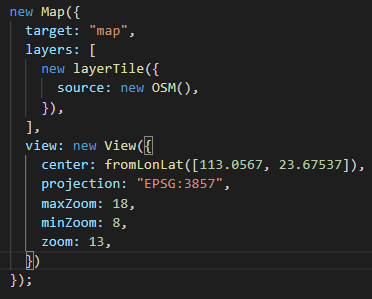
若不进行转换，那么[113.0567, 23.67537]这个位置，则对应地图上某个海上。



要区域正常显示，则需要使用fromLonLat方法，把[113.0567, 23.67537]这个经纬度坐标转为3857平面坐标（单位为米）

fromLonLat([113.0567, 23.67537])转换后值为： [12585414.274767894, 2713900.1300385776]

import { fromLonLat} from "ol/proj";



同理，或许需要把平面坐标系坐标转为经纬度坐标系，则需要调用toLonLat方法即可。

### 自定义投影转换

由于openlayer中默认只有4326和3857两种投影。但是在实际的开发过程中，可能会遇到形形色色各种类型的投影坐标系。那么就需要将其他类型的投影坐标转换为4326或者3857。

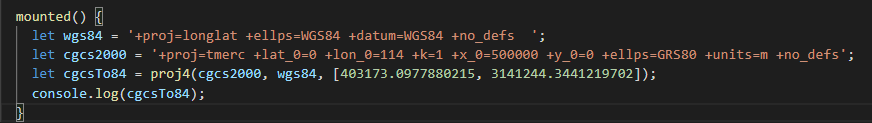
目前推荐使用Proj4对非默认的图标进行转换

Proj4js 是一个开源的 JavaScript 库，用于将点坐标从一个坐标系转换到另一个坐标系。可以使用npm install proj4下载。

使用参考：

import proj4 from "proj4";

把大地2000的某个坐标转为84坐标，



最后得出的结果为：[113.0121206242827, 28.382645741187506]

## 坐标经纬度转换

### 1.**坐标系**类型及转换公式

在进行地图开发过程中，我们一般能接触到以下三种类型的地图坐标系：

1. **WGS-84原始坐标系**，国际标准。Google Earth使用、或者GPS模块

一般用国际GPS纪录仪记录下来的经纬度，通过GPS定位拿到的原始经纬度，Google和高德地图定位的的经纬度（国外）都是基于WGS－84坐标系的；但是在国内是不允许直接用WGS84坐标系标注的，必须经过加密后才能使用；

1. **GCJ-02坐标系(**火星坐标系)，中国坐标偏移标准，Google Map、高德、腾讯使用

是我国国测局独创的坐标体系，由WGS－84加密而成，在国内，必须至少使用GCJ－02坐标系，或者使用在GCJ－02加密后再进行加密的坐标系，如百度坐标系。高德和Google在国内都是使用GCJ－02坐标系，可以说，GCJ－02是国内最广泛使用的坐标系；

1. **百度坐标系：bd-09，**百度地图使用

百度坐标系是在GCJ－02坐标系的基础上再次加密偏移后形成的坐标系，只适用于百度地图。(目前百度API提供了从其它坐标系转换为百度坐标系的API，但却没有从百度坐标系转为其他坐标系的API)

1. **CGCS2000，2000国家大地坐标系**

我们其实很多时候直接用WGS84的坐标来代替CGCS2000坐标。因为CGCS2000的定义与WGS84实质一样。采用的参考椭球非常接近。扁率差异引起椭球面上的纬度和高度变化最大达0.1mm。当前测量精度范围内，可以忽略这点差异。

具体的转换公式，请**[查看附件](地图相关脚本.js)**

### 2.针对项目项目实际情况分析

在项目开发过程中，我们经常需要根据项目情况判断点位数据是否需要纠偏。目前接触的项目，基本都是需要进行纠偏处理。

因为目前我们使用的地图底图基本都是谷歌/高德地图瓦片，坐标系都是使用**火星坐标系**。但是我们在数据库存储的数据，都是用存储**WGS-84原始坐标系**。因此在进行地图操作的时候，需要根据实际情况进行坐标偏移转换。

总结：

数据显示到页面地图，需要进行“84转火星”；

页面数据保存到后台，需要进行“火星转84”

#### (1)“84坐标=>火星”情况

1、在地图点位显示的时候，需要进行“84坐标=>火星”的偏移处理

2、搜索设备，并定位到地图上面

#### (2)“火星=>84坐标”情况

1. 根据当前视野查询视野范围内的设备
2. 地图设备落地，新增设备，保存经纬度信息

## 地图常见操作

### 聚合设备

### 标注

### 热力图

## 常见bug处理