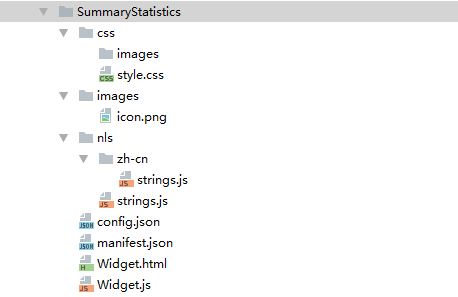
**部分总结**

前言:以下内容或有不正确之处实则不可避免并概不负责亦望包涵并更正

1. **模块化**
2. 1模块文件结构



些许说明：

* Widget.js模块js代码
* Widget.html模块html代码
* css/style.css模块css代码
* images/icon.png是模块默认的图标路径【可配置】
* config.json模块配置文件，比如配置表明，字段等
* manifest.json模块相关配置如：hasConfig【是否有config.json配置】,inPanel【是否在Widget窗口中】等
* nls/zh-cn/strings.js文件主要配置Widget.html中一些固定文本信息

注意：这些文件之所以能做到上面对应效果均基于

hugegis/BaseWidget.js文件【区分’digit/\_WidgetBase’,dojo提供的一个类】

并且Widget.js文件会继承该文件的一些方法。Widget.js文件中几乎都引用hugegis/BaseWidget.js文件，当然只是几乎，具体如何引用文件参考项目代码

1.2一些方法

* 头文件
  + dojo/topic
* 广播与订阅

1. topic.publish("eventName", params);//广播【params传递的参数】
2. topic.subscribe("eventName",hamdlerFunction);//订阅【调用的函数】
3. **var** url = require.toUrl('hugegis');//js文件中多数情况下需要用这样的方式获取本地图片地址

[访问dojo toolkit官网](http://dojotoolkit.org/) [访问arcigs api 3.2官网](https://developers.arcgis.com/javascript/3/jsapi/)

1. **地图交互**
2. 0.在对地图操作之前

* 头文件
  + esri/layers/GraphicsLayer
  + esri/graphic

1. **var** layer = **new** GraphicLayer();
2. map.addLayer(layer);//Widget会自动继承map

注意：以下内容默认已经有以上头文件并实例化layer

2.1画点

* 头文件
  + esri/geometry/Point
  + esri/symbols/PictureMarkerSymbol
* 代码

1. **var** pt = **new** Point(x,y,map.spatialReference);//x,y坐标及参考系
2. **var** pms = **new** PictureMarkerSymbol(url,36,36);//url：point的symbol
3. **var** gra = **new** Graphic(pt,pms);//Graphic也可能是graphic
4. layer.add(gra);

2.2画线

* + 头文件
  + esri/geometry/Polyline
  + esri/symbols/SimpleLineSymbol
  + esri/symbols/SimpleFillSymbol
* 代码

1. //坐标数组
2. **var** points =[];
3. points.push([x,y]);//注意数据组成
5. **var** polyline = **new** Polyline(map.spatialReference);
6. polyline.paths = [points];//数据组成需要重视
7. **var** sls = **new** SimpleLineSymbol(SimpleLineSymbol.STYLE\_SOLID, c, 5);
8. **var** gra = **new** Graphic(**this**.polyline, sls);
9. layer.add(gra);

2.3画面

* 头文件
  + esri/geometry/Polygon
  + esri/symbols/SimpleFillSymbol
  + esri/symbols/SimpleLineSymbol
* 代码

1. **var** polygon = **new** Polygon(map.spatialReference);
2. polygon.addRings([[-180,-90],[-180,90],[180,90],[180,-90],[-180,-90]]);//数据组成需要重视
3. **var** symbol = **new** SimpleFillSymbol("solid", **new** SimpleLineSymbol("solid", **new** dojo.Color([232, 104, 80]), 2), **new** dojo.Color([232, 104, 80, 0.25]));
4. **var** gra = **new** Graphic(polygon,symbol);
5. layer.add(gra);

2.4定位

* 点定位
  + esri/geometry/Point
* 代码

1. **var** pt = **new** Point(x,y,map.spatialReference);//这里一定要是当前地图的空间参考系
2. map.centerAt(pt);//或者map.centerAndZoom(pt,6);

* 面定位

1. **var** extent = polygon.getExtent();//默认已经存在有一个面或者线对象
2. map.setExtent(extent);//map.setExtent(extent.expand(1.5));

2.5 buffer

* 头文件
  + esri/tasks/GeometryService
  + esri/tasks/BufferParameters
  + esri/symbols/SimpleFillSymbol
  + esri/symbols/SimpleLineSymbol
  + esri/geometry/Polygon
  + esri/tasks/AreasAndLengthsParameters
* 代码（第一种：只是处理面的拐角出，变得圆滑）

1. var gs = **new** GeometryService(gsUrl);//appConfig.geometryService地图服务地址
2. var map = this.map;
3. **function** doConvexHull(points) {//points点数组
4. **var** symbol = **new** SimpleFillSymbol("solid", **new** SimpleLineSymbol("solid", **new** dojo.Color([232, 104, 80]), 2), **new** dojo.Color([232, 104, 80, 0.25]));
5. //convexHull
6. gs.convexHull(points).then(**function** (result) {
7. **var** gra = **new** Graphic(result);
8. //buffer
9. **var** params = **new** BufferParameters();
10. params.distances = [parseInt(10)];//bufferDistance距离
11. params.bufferSpatialReference = map.spatialReference;
12. params.geometries = [gra.geometry];
13. gs.buffer(params).then(**function** (buf) {
14. **var** obj = buf;
15. //画面
16. **var** arrPoint = obj[0].rings[0].slice(0);
17. **var** newPolygon = **new** Polygon(arrPoint);
18. newPolygon.setSpatialReference(map.spatialReference);
19. **var** polygonExtent = newPolygon.getExtent();
20. **var** graTemp = **new** Graphic(obj[0]);
21. graTemp.symbol = symbol;
22. layer.add(graTemp);
23. map.setExtent(polygonExtent);
24. //計算面積
25. **var** areasAndLengthsParameters = **new** AreasAndLengthsParameters();
26. areasAndLengthsParameters.polygons = [graTemp.geometry];
27. gs.areasAndLengths(areasAndLengthsParameters, **function** (evt) {
28. **var** obj = evt;
29. });
30. });
31. });
32. },

* 头文件
  + esri/geometry/normalizeUtils
* 代码（第二种：在面的边缘等距离的做buffer，与第一种有极大区别）

1. var gs = **new** GeometryService(gsUrl);//appConfig.geometryService地图服务地址
2. var map = this.map;
3. **function** doConvexHull(geom) {//geom是一个polygon或者是polyline
4. **var** symbol = **new** SimpleFillSymbol("solid", **new** SimpleLineSymbol("solid", **new** dojo.Color([232, 104, 80]), 2), **new** dojo.Color([232, 104, 80, 0.25]));
5. //gs.convexHull 与 normalizeUtils两种buffer的差异
6. normalizeUtils.normalizeCentralMeridian([geom]).then(**function**(result){
7. **var** normalizedGeometry = result[0];
8. //buffer
9. **var** params = **new** BufferParameters();
10. params.distances = [parseInt(10)];//距离
11. params.bufferSpatialReference = map.spatialReference;
12. params.geometries = [normalizedGeometry];
13. gs.buffer(params).then(**function** (buf) {
14. **var** obj = buf;
15. **var** arrPoint = obj[0].rings[0].slice(0);
16. //转换成esri对象，求取Extent
17. **var** newPolygon = **new** Polygon(arrPoint);
18. newPolygon.setSpatialReference(map.spatialReference);
19. **var** polygonExtent = newPolygon.getExtent();
20. //绘制
21. **var** gra = **new** Graphic(obj[0]);
22. gra.symbol = symbol;
23. layer.add(gra);
24. map.setExtent(polygonExtent);
25. });
26. });
27. },

2.6画图工具

* 头文件
  + esri/toolbars/draw
  + esri/symbols/PictureMarkerSymbol
* 代码

1. **var** map = **this**.map;
2. **var** symbol  = **new** PictureMarkerSymbol(url, 36, 36);
3. **var** drawToolBar = **new** Draw(map);
4. drawToolBar.setMarkerSymbol(symbol);
5. on(drawToolBar, 'draw-end', onDrawEnd);//onDrawEnd画图完毕回调函数
6. drawToolBar.activate(Draw.POINT);//激活画点，其他如画面，画线可参考arcgis api,方式几乎一样
8. **function** onDrawEnd(e){
9. **var** geo = e.geometry;
10. **var** gra = **new** Graphic(geo,symbol);
11. layer.add(gra);
12. drawToolBar.deactivate();
13. }