# ArcGIS学习笔记

# 一、ArcGIS

ArcGIS是Esri公司集40余年地理信息系统（GIS）咨询和研发经验，奉献给用户的一套完整的GIS平台产品，具有强大的地图制作、空间数据管理、空间分析、空间信息整合、发布与共享的能力。

2014年底，ArcGIS 10.3正式发布。ArcGIS 10.3中，以用户为中心（Named User）的全新授权模式，超强的三维"内芯"，革新性的桌面GIS应用，可配置的服务器门户，即拿即用的Apps，更多应用开发新选择，数据开放新潮流，为构建新一代Web GIS应用提供了更强有力的支持。

## 关于 ArcGIS

ArcGIS 为单用户或多用户在桌面、服务器、Web 和野外实现 GIS 提供了可伸缩的框架。ArcGIS 是 GIS 软件产品的集成系列，这些产品构成了一个完整的 GIS。ArcGIS 由几个用于部署 GIS 的主要框架组成：

**ArcGIS Desktop -**一套集成的专业 GIS 应用程序。大多数用户将其视为由以下三种产品组成：ArcView、ArcEditor 和 ArcInfo。

**ArcGIS Server -**将 GIS 信息和地图以 Web 服务形式发布，提供一系列 Web GIS 应用程序，并且支持企业级数据管理。

**ArcGIS Mobile -**为野外计算提供移动 GIS 工具和应用程序。

**ArcGIS Online -**提供可通过 Web 进行访问的在线 GIS 功能，外加 ESRI 与合作伙伴发布的可供用户在自己的 Web GIS 应用程序中使用的有用地图和数据。

**ArcGIS Engine -**为使用 C++、.NET 或 Java 的 ArcGIS 开发人员提供软件组件库。

## 1、ArcGIS Online

ArcGIS Online 是基于云的协作式平台，允许组织成员使用、创建和共享地图、应用程序和数据，以及访问权威性底图和 ArcGIS 应用程序。通过 ArcGIS Online，您可以访问 Esri 的安全云，在其中可将数据作为发布的 web 图层进行管理、创建和存储。由于 ArcGIS Online 是 ArcGIS 系统的组成部分，您还可以利用其扩展 ArcGIS for Desktop、ArcGIS for Server、ArcGIS Web API 和 ArcGIS Runtime SDK 的功能。

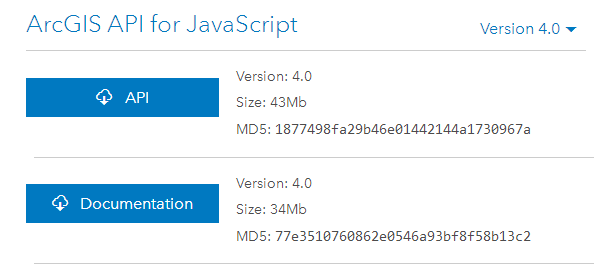
## 2、ArcGIS for Server

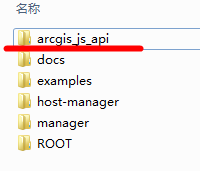
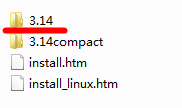
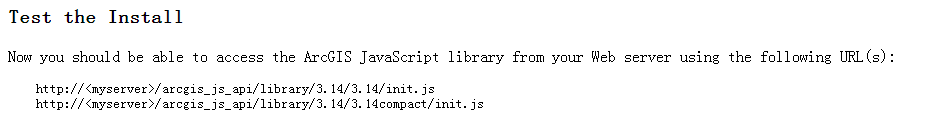
ArcGIS for Server是基于服务器的 ArcGIS 工具，通过Web Services在网络上提供GIS资源和功能服务，其发布的 GIS 服务遵循广泛采用的 Web 访问和使用标准。ArcGIS for Server 广泛用于企业级 GIS 实现以及各种 Web GIS 应用程序中，不但可以在本地还可以在云基础设施上配置,运行于Windows 及 Linux 服务器环境。

## 3、ArcGIS for Desktop

ArcGIS for Desktop是为GIS专业人士提供的用于信息制作和使用的工具，利用它可以实现任何从简单到复杂的GIS任务。ArcGIS for Desktop的功能特色主要包括：高级的地理分析和处理能力、提供强大的编辑工具、拥有完整的地图生产过程，以及无限的数据和地图分享体验。

## 4、ArcGIS API for JavaScript3.14本地部署(tomcat)

1.ArcGIS API for **[JavaScript](http://lib.csdn.net/base/javascript" \o "JavaScript知识库" \t "http://blog.csdn.net/qq_31598891/article/details/_blank)**下载地址：   
<https://developers.arcgis.com/downloads/>   


2.解压API文件，并将其中arcgis\_js\_api文件复制到tomcat安装目录下的webapps中，本文路径如下：   
C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps   
   
3.文件配置：   
打开arcgis\_js\_api\library\3.14，有3.14和3.14compact两个版本的文件，只需要对其中一个版本文件进行配置即可。   
   
一共有两处需要进行配置。打开3.14文件夹中的init.js,全文搜索将文本 ‘[HOSTNAME\_AND\_PATH\_TO\_JSAPI]’替换成如下：   
< myserver >/arcgis\_js\_api/library/3.14/3.14/   
再打开arcgis\_js\_api\library\3.14\3.14\dojo\dojo.js，同样进行上述处理。   
**注意：**   
在此使用的是本地服务器，myserver设置为127.0.0.1:8080,要加上端口号。   
4.开启tomcat，测试文件配置：   
  
输入网址127.0.0.1:8080//arcgis\_js\_api/library/3.14/3.14/init.js，显示如下即为文件配置成功：   
  
5.测试安装成功：   
在这里可以加载一段代码显示地图

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"><html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

<title>Simple Map</title>

<link rel="stylesheet" type="text/css" href="http://127.0.0.1:8080/arcgis\_js\_api/library/3.14/3.14/dijit/themes/tundra/tundra.css"/>

<link rel="stylesheet" type="text/css" href="http://127.0.0.1:8080/arcgis\_js\_api/library/3.14/3.14/esri/css/esri.css" />

<script type="text/javascript" src="http://127.0.0.1:8080/arcgis\_js\_api/library/3.14/3.14/init.js"></script>

<script type="text/javascript">

dojo.require("esri.map");

function init() {

var myMap = new esri.Map("mapDiv");

//note that if you do not have public Internet access then you will need to point this url to your own locally accessible cached service.

var myTiledMapServiceLayer = new esri.layers.ArcGISTiledMapServiceLayer("http://server.arcgisonline.com/ArcGIS/rest/services/NGS\_Topo\_US\_2D/MapServer");

myMap.addLayer(myTiledMapServiceLayer);

}

dojo.addOnLoad(init);

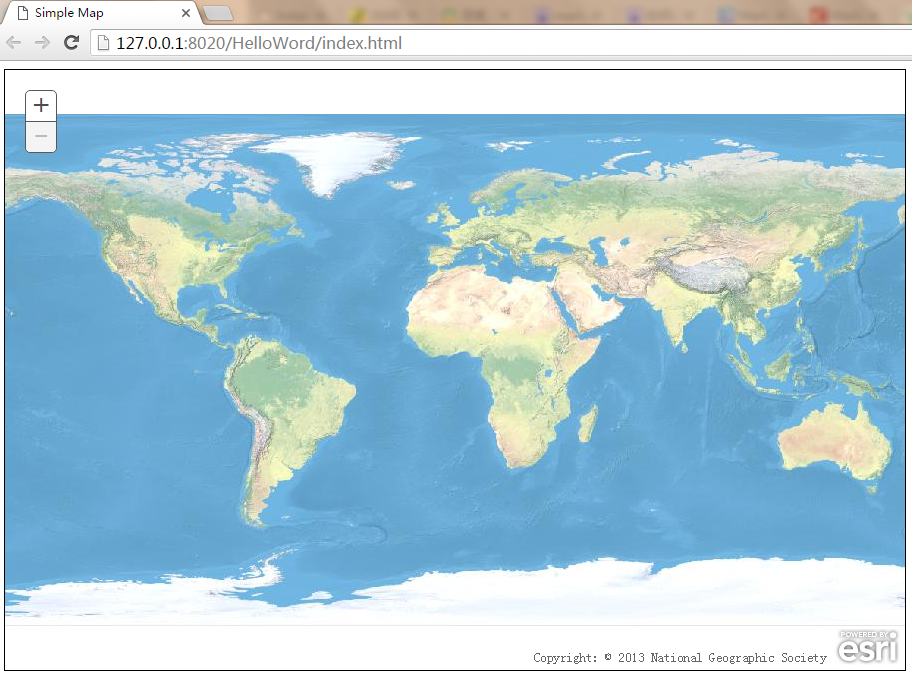
</script>

</head>

<body class="tundra">

<div id="mapDiv" style="width:900px; height:600px; border:1px solid #000;"></div>

</body></html>

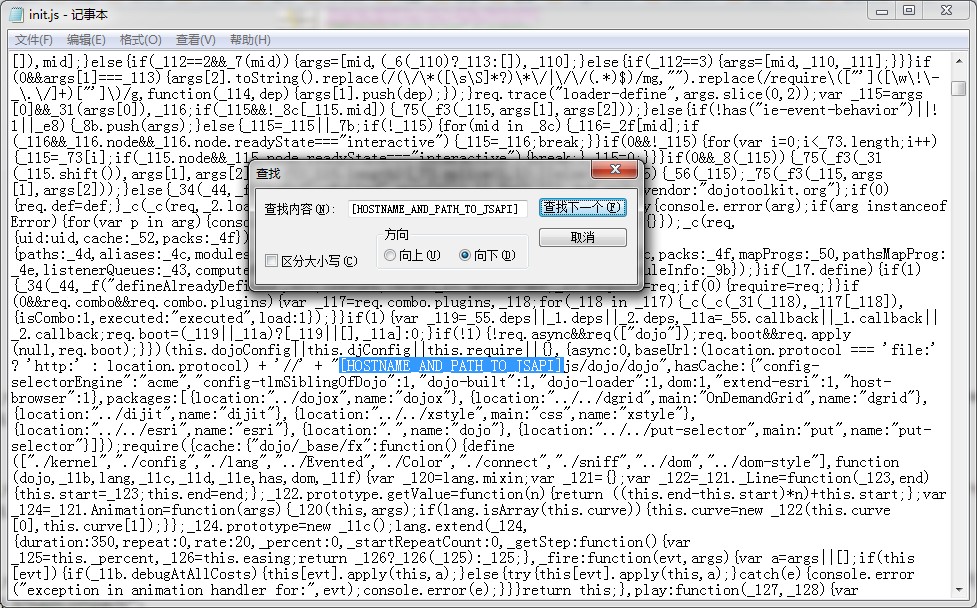
在浏览器中成功显示   


-----------------------部署API方法---------------------

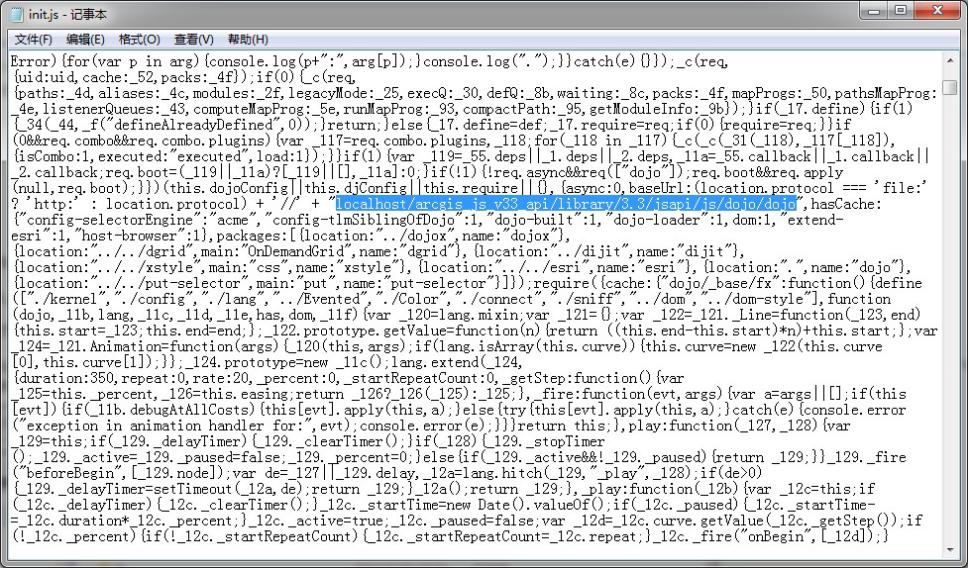
1、解压arcgis\_js\_v33\_sdk.zip和arcgis\_js\_v33\_api.zip；

　　2、进行配置文件的修改，使其被其他程序调用时找到引用函数的位置；

　　(1)找到..\arcgis\_js\_v33\_api\library\3.3\jsapi\init.js文件，打开，可以通过EditPlus打开；查找'[HOSTNAME\_AND\_PATH\_TO\_JSAPI]'，替换为"<myserver>/ \arcgis\_js\_v33\_api\library\3.3\jsapi/"；其中<myserver>为机器名称或者机器IP，没有http前缀（如果在本地使用可以设为localhost）；



**改变前效果图**



**改变后效果图(以下不再截图，与该类似)**

　　(2)找到..\arcgis\_js\_v33\_api\library\3.3\jsapi\js\dojo\dojo\dojo.js文件，打开；查找'[HOSTNAME\_AND\_PATH\_TO\_JSAPI]'，替换为"<myserver>/ arcgis\_js\_v33\_api/library/3.3/jsapi/"；

　　(3)找到..\arcgis\_js\_v33\_api\library\3.3\jsapicompact\init.js文件，打开；查找'[HOSTNAME\_AND\_PATH\_TO\_JSAPI]'，替换为"<myserver>/ arcgis\_js\_v33\_api/library/3.3/jsapicompact/"；

　　(4)找到..\arcgis\_js\_v33\_api\library\3.3\jsapicompact\js\dojo\dojo\dojo.js文件，打开；查找'[HOSTNAME\_AND\_PATH\_TO\_JSAPI]'，替换为"<myserver>/ arcgis\_js\_v33\_api/library/3.3/jsapicompact/"；

　　保存之后，将arcgis\_js\_v33\_sdk文件夹直接拷贝到C:\Inetpub\wwwroot下，变成C:\Inetpub\wwwroot\arcgis\_js\_v33\_sdk；

      将arcgis\_js\_v33\_api文件夹直接拷贝到C:\Inetpub\wwwroot下，变成C:\inetpub\wwwroot\arcgis\_js\_v33\_api；

# 二、ArcGIS API for Javascript

ArcGIS API for Javascript可以在多种不同的设备上使用，任何屏幕，任何浏览器上都可以使用ArcGIS API for Javascript 开发的地图应用。ArcGIS API for Javascript利用最新的HTML 5和CSS 3标准使你的地图应用程序的灵活性和性能大大增加。10.3版本中Esri将推出全新的基于JavaScript API的Web App Builder，用户通过它可以像使用Flex和Silverlight Viewers无需编码，配置基于JavaScript的web应用程序。

**ArcGIS API for Javascript主要功能**

• 空间数据展示：加载地图服务，影像服务,WMS等。

• 客户端 Mashup：将来自不同服务器、不同类型的服务在客户端聚合后统一呈现给客户。

• 图形绘制：在地图上交互式地绘制查询范围或地理标记等。

• 符号渲染：提供对图形进行符号化,要素图层生成专题图和服务器端渲染等功能。

• 查询检索：基于属性和空间位置进行查询，支持关联查询，对查询结果的排序、分组以及对属性数据的统计。

• 地理处理：调用 ArcGIS for Server 发布的地理处理服务（GP 服务），执行空间分析、地理处理或其他需要服务器端执行的工具、模型、运算等。

• 网络分析：计算最优路径、临近设施和服务区域。

• 在线编辑：通过要素服务编辑要素的图形、属性、附件,进行编辑追踪。

• 时态感知：展示、查询具有时间特征的地图服务或影像服务数据。

• 影像处理：提供动态镶嵌、实时栅格函数处理等功能。

• 地图输出：提供多种地图图片导出和服务器端打印等功能。

• Web App Builder

Web App Builder 是Esri推出的基于ArcGIS API for Javascript用于帮助用户快速开发Web GIS 系统的快速开发模板，该模板的特点如下：

可创建2、3维度的Web app

可选的丰富可配置的widget 库

• 提供多种部件库，开发者可根据需要灵活选择，部件包括Chart、Draw、Legend、MapCompare、Search、Query等等。

灵活的配置

• 开发者可灵活配置web应用的界面布局，主题风格等。

响应式的界面设计

• Web AppBuilder采用了响应式界面设计，配置出的应用可以适应多种尺寸的屏幕，支持移动端。

配置的应用可共享到云端

• 可以将应用发布到Online上作为web App可配置的模板。

支持灵活扩展

• 开发者可以方便的基于JSAPI开发出自定义的Widget。

**ArcGIS API for Javascript 新功能**

• 可以通过HeatmapRenderer更新FeatureLayer的热点渲染

• 增加了10个新的Widget

• [CalculateDensity](https://developers.arcgis.com/javascript/jsapi/calculatedensity-amd.html" \t "http://www.esrichina.com.cn/2015/0108/_blank) – 根据点或者线要素创建密度图

• [ConnectOriginsToDestinations](https://developers.arcgis.com/javascript/jsapi/connectoriginstodestinations-amd.html" \t "http://www.esrichina.com.cn/2015/0108/_blank) – 计算多组点之间的距离和通行时间

• [CreateViewshed](https://developers.arcgis.com/javascript/jsapi/createviewshed-amd.html" \t "http://www.esrichina.com.cn/2015/0108/_blank) –计算可视分析.

• [CreateWatersheds](https://developers.arcgis.com/javascript/jsapi/createwatersheds-amd.html" \t "http://www.esrichina.com.cn/2015/0108/_blank) – 根据指定的位置计算流域

• [DeriveNewLocations](https://developers.arcgis.com/javascript/jsapi/derivenewlocations-amd.html" \t "http://www.esrichina.com.cn/2015/0108/_blank) -根据指定的条件获得新的位置。

• [FindExistingLocations](https://developers.arcgis.com/javascript/jsapi/findexistinglocations-amd.html" \t "http://www.esrichina.com.cn/2015/0108/_blank) – 根据属性查询和空间查询条件查找存在的位置

[FindSimilarLocations](https://developers.arcgis.com/javascript/jsapi/findsimilarlocations-amd.html" \t "http://www.esrichina.com.cn/2015/0108/_blank) -根据一个或者多个参考位置计算相近的位置。

• [InterpolatePoints](https://developers.arcgis.com/javascript/jsapi/interpolatepoints-amd.html" \t "http://www.esrichina.com.cn/2015/0108/_blank) – 根据已有的测量点进行点的内插.

• [PlanRoutes](https://developers.arcgis.com/javascript/jsapi/planroutes-amd.html" \t "http://www.esrichina.com.cn/2015/0108/_blank) -决定在使用mobile的员工之间如何有效的分配任务

• [TraceDownstream](https://developers.arcgis.com/javascript/jsapi/tracedownstream-amd.html" \t "http://www.esrichina.com.cn/2015/0108/_blank) – 根据指定的位置计算下游追中分析

• 提供标注能力

• to/from MGRS 等

几何服务增加了两个新的方法，这两个方法很容易的实现 to/from MGRS, USNG, UTM 等的转换。

## 地图属性

### 1) baseMap Properties

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Summary** |
| [dark-gray](https://developers.arcgis.com/javascript/3/jsapi/esri.basemaps-amd.html" \l "dark-gray) | [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) | The Dark Gray Canvas basemap is designed to be used as a soothing background map for overlaying and focus attention on other map layers. |
| [dark-gray-vector](https://developers.arcgis.com/javascript/3/jsapi/esri.basemaps-amd.html" \l "dark-gray-vector) | [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) | This vector tile layer provides a detailed basemap for the world featuring a neutral background style with minimal colors, labels, and features. |
| [gray](https://developers.arcgis.com/javascript/3/jsapi/esri.basemaps-amd.html" \l "gray) | [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) | The Light Gray Canvas basemap is designed to be used as a neutral background map for overlaying and emphasizing other map layers. |
| [gray-vector](https://developers.arcgis.com/javascript/3/jsapi/esri.basemaps-amd.html" \l "gray-vector) | [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) | This vector tile layer provides a detailed basemap for the world featuring a neutral background style with minimal colors, labels, and features. |
| [hybrid](https://developers.arcgis.com/javascript/3/jsapi/esri.basemaps-amd.html" \l "hybrid) | [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) | The World Imagery map is a detailed imagery map layer and labels that is designed to be used as a basemap for various maps and applications. |
| [national-geographic](https://developers.arcgis.com/javascript/3/jsapi/esri.basemaps-amd.html" \l "national-geographic) | [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) | The National Geographic basemap is designed to be used as a general reference map for informational and educational purposes. |
| [oceans](https://developers.arcgis.com/javascript/3/jsapi/esri.basemaps-amd.html" \l "oceans) | [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) | The Ocean Basemap is designed to be used as a basemap by marine GIS professionals and as a reference map by anyone interested in ocean data. |
| [osm](https://developers.arcgis.com/javascript/3/jsapi/esri.basemaps-amd.html" \l "osm) | [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) | The OpenStreetMap is a community map layer that is designed to be used as a basemap for various maps and applications. |
| [satellite](https://developers.arcgis.com/javascript/3/jsapi/esri.basemaps-amd.html" \l "satellite) | [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) | The World Imagery map is a detailed imagery map layer that is designed to be used as a basemap for various maps and applications. |
| [streets](https://developers.arcgis.com/javascript/3/jsapi/esri.basemaps-amd.html" \l "streets) | [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) | The Streets basemap presents a multiscale street map for the world. |
| [streets-navigation-vector](https://developers.arcgis.com/javascript/3/jsapi/esri.basemaps-amd.html" \l "streets-navigation-vector) | [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) | This vector tile layer provides a detailed basemap for the world featuring a custom navigation map style. |
| [streets-night-vector](https://developers.arcgis.com/javascript/3/jsapi/esri.basemaps-amd.html" \l "streets-night-vector) | [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) | This vector tile layer provides a detailed basemap for the world featuring a custom "night time" street map style. |
| [streets-relief-vector](https://developers.arcgis.com/javascript/3/jsapi/esri.basemaps-amd.html" \l "streets-relief-vector) | [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) | This vector tile layer provides a detailed basemap for the world featuring a classic Esri street map style designed for use with a relief map. |
| [streets-vector](https://developers.arcgis.com/javascript/3/jsapi/esri.basemaps-amd.html" \l "streets-vector) | [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) | This vector tile layer provides a detailed basemap for the world featuring a classic Esri street map style. |
| [terrain](https://developers.arcgis.com/javascript/3/jsapi/esri.basemaps-amd.html" \l "terrain) | [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) | The Terrain with Labels basemap is designed to be used to overlay and emphasize other thematic map layers. |
| [topo](https://developers.arcgis.com/javascript/3/jsapi/esri.basemaps-amd.html" \l "topo) | [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) | The Topographic map includes boundaries, cities, water features, physiographic features, parks, landmarks, transportation, and buildings. |
| [topo-vector](https://developers.arcgis.com/javascript/3/jsapi/esri.basemaps-amd.html" \l "topo-vector) | [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) | This vector tile layer provides a detailed basemap for the world featuring a classic Esri topographic map style designed for use with a relief map. |

### 2) baseMap code

<!DOCTYPE html>

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"*>

<meta name=*"viewport"* content=*"initial-scale=1, maximum-scale=1,user-scalable=no"*/>

<title>Simple Map</title>

<link rel=*"stylesheet"* href=*"https://js.arcgis.com/3.18/esri/css/esri.css"*>

<style>

**html,** **body,** *#map* {

height: *100%*;

margin: *0*;

padding: *0*;

}

*#BasemapToggle* {

position: *absolute*;

top: *20px*;

right: *20px*;

z-index: *50*;

}

</style>

<script src=*"https://js.arcgis.com/3.18/"*></script>

<script>

**var** map;

require(["esri/map",

"esri/dijit/BasemapToggle",

"esri/layers/ArcGISDynamicMapServiceLayer",

"esri/layers/ImageParameters",

"dojo/domReady!"], **function**(Map,BasemapToggle,ArcGISDynamicMapServiceLayer,ImageParameters) {

map = **new** Map("map", {

basemap: "topo",

center: [120.15, 30.26], // longitude, latitude

zoom: 13

});

**var** toggle = **new** BasemapToggle({

map: map,

basemap: "satellite"

}, "BasemapToggle");

toggle.startup();

});

</script>

</head>

<body>

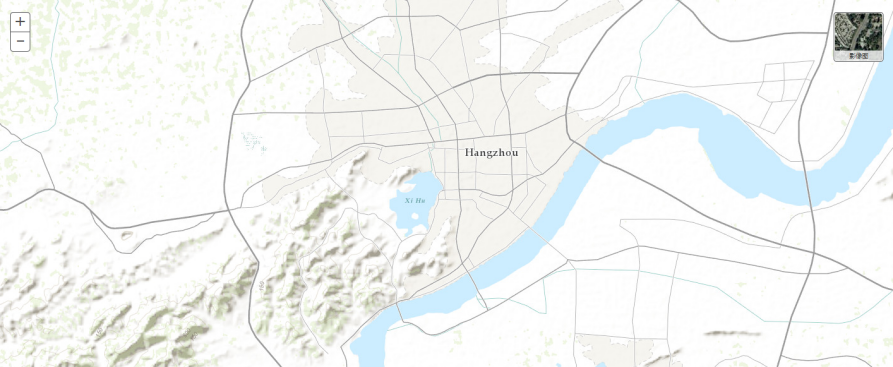
<div id=*"map"*>

<div id=*"BasemapToggle"*></div>

</div>

</body>

</html>



### 3) Map 简介

Arcgis for Javasctipt中常见的layer有

动态图层（ArcGISDynamicMapServiceLayer）、

切片图层（ArcGISTiledMapServiceLayer）、

特征图层（FeatureLayer）、

图象图层（GraphicsLayer）、

标注图层（LabelLayer）、

wms图层（WMSLayer）

和切片wms图层（WMTSLayer）等几种。

ARCGis知乎：

<http://zhihu.esrichina.com.cn/article/556>

### 4) Map方法描述

矢量图：ArcGISDynamicMapServiceLayer

影像图：ArcGISImageServiceLayer

//创建graphicsLayer

**var** graphicLines=**new** esri.layers.GraphicsLayer();

var graphicLines=new esri.layers.GraphicsLayer({id:”graFun”});

//添加到地图

map.addLayer(graphicLines)

#### GraphicsLayer添Graphic(点、线、面)

前提:**从feature创建Graphic.**

**如果无feature**,请参考从坐标生成feature.

步骤:为feature设置symbol

关键句: var gra=new esri.Graphic(feature,symbol,attributes,infoTemplate);

       graphicsLayer.add(gra);

备注:-1.graphic的feature是传址引用。

-2.如果需要从坐标生成feature,参考feature操作。

#### 设置点symbol,生成 graphic

var graphicLayer=new esri.layers.GraphicsLayer();

//方法1

feature.setSymbol(**new** esri.symbol.PictureMarkerSymbol(“start.png”, 21, 21));

feature.setInfoTemplate("起点信息", "${NAME}");

graphicLayer.add(feature);

//方法2

var pointGra=new esri.Graphic(feature,symbol,attributes,infoTemplate);

graphicLayer.add(pointGra);

备注:创建graphic,feature是必须的.

必须有symbol才能显示.

attributes，可选。

infoTemplate,点击graphic弹出的信息窗体,可选.

#### 设置面symbol,生成graphic

**var** ringPolygons=**new** esri.layers.GraphicsLayer();

//方法1：

polygonFeature.setSymbol(**new** esri.symbol.SimpleFillSymbol(esri.symbol.SimpleFillSymbol.STYLE\_SOLID, **new**esri.symbol.SimpleLineSymbol(esri.symbol.SimpleLineSymbol.STYLE\_DASHDOT, **new** dojo.Color([255, 0, 0]), 2), **new** dojo.Color(colors.Yellow)));

graphicsLayer.add(polygonFeature);

方法2：

var graPolygon=new esri.Graphic(polygonFeature,symbol,null,null)

graphicsLayer.add(graPolygon);

#### point操作：

**坐标生成point**

**var** point=**new** esri.geometry.Point(x,y,{wkid:102113});

var point= new esri.geometry.Point(-118.15, 33.80, newesri.SpatialReference({ wkid: 4326 }));

**复制point feature**

**this**.getCopyOfTransfers=**function**(){

**var** results=[];

**for**(**var** i=0;i<=pointFeatures.length-1;i++){

**var** geoJson= pointFeatures [i].geometry.toJson();

**var** attrJson=JSON.stringify(pointFeatures [i].attributes);

**var** newAttr=JSON.parse(attrJson);

            newAttr.odType="w";

**var** newTransfer=**new** esri.Graphic(**new** esri.geometry.Point(geoJson),**null**,newAttr,**null**);

            results.push(newTransfer);

         }

**return** results;

   };

#### feature系列

**feature=geometry+attributes+infoTemplate+symbol**

feature is **equivalent** to graphic.

 geometry存矢量信息

 attributes存属性信息

 infoTemplate 弹出窗体

 symbol  feature的符号

**feature.setInfoTemplate**

//弹出窗体的 标题和内容

  middlefeature.setInfoTemplate(new esri.InfoTemplate(this.title, "<tr>途经换乘:<td>${routeName}</tr></td><br><tr>点到点OD总

量:<td>${ODTotal}</tr></td><br><tr>本方案OD量:<td>${ODValue}</tr></td><br><tr>占比:<td>${ODPartition}</tr></td>"));

### 6) Map操作

#### map组件

说明:new esri.Map(div,{options}); map组件指options

   mapObj=**new** esri.Map("map",{

      nav:**true**,//8个pan 箭头

      slider:**false**,//左上的缩放 +/-;

      logo:**false**,//右下的esri logo

      showAttribution:**false**,//右下的gisNeu (logo左侧)

      extent://地图的extent

      });

控制ZoomSlider的方法

(1)map.showZoomSlider();

   (2)map.hideZoomSlider();

控制nav的方法

  (1)map.showPanArrows();

  (2)map.hidePanArrows();

执行nav8个方向移动的方法

  .panUpperLeft()

  .panUp();

  .panUpperRight()

  .panLeft()

  .panRight()

  .panLowerLeft();

  .panDown()

  .panLowerRight();

#### 鼠标移动 显示坐标

Code snippets:

  dojo.connect(map, "onMouseMove", showCoordinates);

   function showCoordinates(evt) {

        //get mapPoint from event

        //The map is in web mercator - modify the map point to display the results in geographic

        var mp = esri.geometry.webMercatorToGeographic(evt.mapPoint);

        //display mouse coordinates

        dojo.byId("info").innerHTML = mp.x.toFixed(3) + ", " + mp.y.toFixed(3);

      }

#### 鼠标事件 down/up/drag

支持：mouseDown,mouseUp

不支持：mouseMove,mouseDrag

//surpport in 3.4

      dojo.connect(map,"onMouseDown",**function**(evt){

         console.log("mouseDown triggered");

      });

      //sur int 3.4

      dojo.connect(map,"onMouseUp",**function**(evt){

         console.log("mouseUp triggered");

      });

      //not sur in 3.4

      dojo.connect(map,"onMouseMove",**function**(evt){

         console.log("mouseMove triggered");

      });

      dojo.connect(map,"onMouseDragStart",**function**(evt){

          console.log("dragStart trigered");

         startPt=evt;

          console.log("drawStart triggered,start point:"+evt.mapPoint.x+","+evt.mapPoint.y+".screenPoint:"+evt.clientX+","+evt.clientY);

       });

      dojo.connect(map,"onMouseDrag",**function**(evt){

         console.log("mouseDrag triggered");

      });

      dojo.connect(map,"onMouseDragEnd",**function**(evt){

         console.log("mouseDragEnd triggered");

      });

#### map. setMapCursor 设置地图指针(鼠标样式)

map.setMapCursor("help") //

map.setMapCursor("default")//

map.setMapCursor("pointer")//

map.setMapCursor("wait")//

map.setMapCursor("progress")//

map.setMapCursor("cell")// 粗十字

map.setMapCursor("crosshair")// 细十字

map.setMapCursor("text")//文本I

map.setMapCursor("vertical-text")//放倒的I

### 7.绘制要素操作

#### 基础类

#### 画完军规

军规：activateDrawTool,必须启动deactivate使鼠标恢复正常

//鼠标画矩形

drawTool.activate(esri.toolbars.Draw.POINT);

//鼠标终止矩形

drawTool.deactivate

#### 点

drawTool.activate(esri.toolbars.Draw.POINT);

#### 线

drawTool.activate(esri.toolbars.Draw.POLYLINE);

#### 矩形

drawTool.activate(esri.toolbars.Draw.EXTENT);

#### 圆

drawTool.activate(esri.toolbars.Draw.CIRCLE);

# **dojo**

**1>dojo不只是一个程序库，而是一个工具包。**这就说明了dojo的完整功能包的体积是相对比较大的，当然，完整包的功能要超出你的想象的强大，strong！

**1.1>**dojo能帮助我们统一浏览器差异，可以让我们的代码在多个主流i浏览器中健壮的运行。不再需要进行浏览器或特性判断，也不会有依赖浏览器的代码。简单点，一句话，就是完美的跨浏览器解决方案。（IE6+）（FireFox）（Safari）（Opera）  
**1.2>**dojo一个函数库，用于解决几个严重的浏览器错误，例如内存泄露。  
**1.3>**一个已知的最丰富的HTML图形组件集  
**1.4>**模块体系连同打包系统，可以让你在开发以及将来发布的过程中将代码拆分成较小的，可管理的分块，以便优化下载性能。这个过程中不需要更改任何代码。打包系统甚至允许将Dojo本身进行拆分，使其适合你的项目。  
**1.5>**独立的模块（按需加载），他们实现了一些高级功能  
**1.6>**另外还有自定义HTML小部件（widget），国家化（internationalization，i18n），本地化（localization ，110n），无障碍（accessible，a11y）  
  
dojo使用纯JavaScript编写，可以用于非浏览器环境，内嵌SpiderMonkey以及内嵌Rhina环境中。  
  
Dojo是开源的，你可以访问 [http://www.dojotoolkit.org/](http://www.dojotoolkit.org/" \t "http://www.cnblogs.com/didi/archive/2010/05/26/_blank) 下载你所需要的dojo包。

# Require.js

## 一、为什么要用require.js？

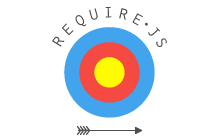
最早的时候，所有Javascript代码都写在一个文件里面，只要加载这一个文件就够了。后来，代码越来越多，一个文件不够了，必须分成多个文件，依次加载。下面的网页代码，相信很多人都见过。

　　<script src="1.js"></script>  
　　<script src="2.js"></script>  
　　<script src="3.js"></script>  
　　<script src="4.js"></script>  
　　<script src="5.js"></script>  
　　<script src="6.js"></script>

这段代码依次加载多个js文件。

这样的写法有很大的缺点。首先，加载的时候，浏览器会停止网页渲染，加载文件越多，网页失去响应的时间就会越长；其次，由于js文件之间存在依赖关系，因此必须严格保证加载顺序（比如上例的1.js要在2.js的前面），依赖性最大的模块一定要放到最后加载，当依赖关系很复杂的时候，代码的编写和维护都会变得困难。

### 1、require.js的诞生，就是为了解决这两个问题：



　　（1）实现js文件的异步加载，避免网页失去响应；

　　（2）管理模块之间的依赖性，便于代码的编写和维护。

## 二、require.js的加载

使用require.js的第一步，是先去官方网站[下载](http://requirejs.org/docs/download.html" \t "http://www.ruanyifeng.com/blog/2012/11/_blank)最新版本。

下载后，假定把它放在js子目录下面，就可以加载了。

　　<script src="js/require.js"></script>

有人可能会想到，加载这个文件，也可能造成网页失去响应。解决办法有两个，一个是把它放在网页底部加载，另一个是写成下面这样：

　　<script src="js/require.js" ****defer async="true"**** ></script>

async属性表明这个文件需要异步加载，避免网页失去响应。IE不支持这个属性，只支持defer，所以把defer也写上。

加载require.js以后，下一步就要加载我们自己的代码了。假定我们自己的代码文件是main.js，也放在js目录下面。那么，只需要写成下面这样就行了：

　　<script src="js/require.js" ****data-main="js/main"****></script>

data-main属性的作用是，指定网页程序的主模块。在上例中，就是js目录下面的main.js，这个文件会第一个被require.js加载。由于require.js默认的文件后缀名是js，所以可以把main.js简写成main。

## 三、主模块的写法

上一节的main.js，我把它称为"主模块"，意思是整个网页的入口代码。它有点像C语言的main()函数，所有代码都从这儿开始运行。

下面就来看，怎么写main.js。

如果我们的代码不依赖任何其他模块，那么可以直接写入javascript代码。

　　// main.js

　　alert("加载成功！");

但这样的话，就没必要使用require.js了。真正常见的情况是，主模块依赖于其他模块，这时就要使用AMD规范定义的的require()函数。

　　// main.js

　　require(['moduleA', 'moduleB', 'moduleC'], function (moduleA, moduleB, moduleC){

　　　　// some code here

　　});

require()函数接受两个参数。第一个参数是一个数组，表示所依赖的模块，上例就是['moduleA', 'moduleB', 'moduleC']，即主模块依赖这三个模块；第二个参数是一个回调函数，当前面指定的模块都加载成功后，它将被调用。加载的模块会以参数形式传入该函数，从而在回调函数内部就可以使用这些模块。

require()异步加载moduleA，moduleB和moduleC，浏览器不会失去响应；它指定的回调函数，只有前面的模块都加载成功后，才会运行，解决了依赖性的问题。

下面，我们看一个实际的例子。

假定主模块依赖jquery、underscore和backbone这三个模块，main.js就可以这样写：

　　require(['jquery', 'underscore', 'backbone'], function ($, \_, Backbone){

　　　　// some code here

　　});

require.js会先加载jQuery、underscore和backbone，然后再运行回调函数。主模块的代码就写在回调函数中。

## 四、模块的加载

上一节最后的示例中，主模块的依赖模块是['jquery', 'underscore', 'backbone']。默认情况下，require.js假定这三个模块与main.js在同一个目录，文件名分别为jquery.js，underscore.js和backbone.js，然后自动加载。

使用require.config()方法，我们可以对模块的加载行为进行自定义。require.config()就写在主模块（main.js）的头部。参数就是一个对象，这个对象的paths属性指定各个模块的加载路径。

　　require.config({

　　　　paths: {

　　　　　　"jquery": "jquery.min",  
　　　　　　"underscore": "underscore.min",  
　　　　　　"backbone": "backbone.min"

　　　　}

　　});

上面的代码给出了三个模块的文件名，路径默认与main.js在同一个目录（js子目录）。如果这些模块在其他目录，比如js/lib目录，则有两种写法。一种是逐一指定路径。

　　require.config({

　　　　paths: {

　　　　　　"jquery": "****lib/****jquery.min",  
　　　　　　"underscore": "****lib/****underscore.min",  
　　　　　　"backbone": "****lib/****backbone.min"

　　　　}

　　});

另一种则是直接改变基目录（baseUrl）。

　　require.config({

****baseUrl: "js/lib",****

　　　　paths: {

　　　　　　"jquery": "jquery.min",  
　　　　　　"underscore": "underscore.min",  
　　　　　　"backbone": "backbone.min"

　　　　}

　　});

如果某个模块在另一台主机上，也可以直接指定它的网址，比如：

　　require.config({

　　　　paths: {

　　　　　　"jquery": "https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min"

　　　　}

　　});

require.js要求，每个模块是一个单独的js文件。这样的话，如果加载多个模块，就会发出多次HTTP请求，会影响网页的加载速度。因此，require.js提供了一个[优化工具](http://requirejs.org/docs/optimization.html" \t "http://www.ruanyifeng.com/blog/2012/11/_blank)，当模块部署完毕以后，可以用这个工具将多个模块合并在一个文件中，减少HTTP请求数。

## 五、AMD模块的写法

require.js加载的模块，采用AMD规范。也就是说，模块必须按照AMD的规定来写。

具体来说，就是模块必须采用特定的define()函数来定义。如果一个模块不依赖其他模块，那么可以直接定义在define()函数之中。

假定现在有一个math.js文件，它定义了一个math模块。那么，math.js就要这样写：

　　// math.js

　　define(function (){

　　　　var add = function (x,y){

　　　　　　return x+y;

　　　　};

　　　　return {

　　　　　　add: add  
　　　　};

　　});

加载方法如下：

　　// main.js

　　require(['math'], function (math){

　　　　alert(math.add(1,1));

　　});

如果这个模块还依赖其他模块，那么define()函数的第一个参数，必须是一个数组，指明该模块的依赖性。

　　define(['myLib'], function(myLib){

　　　　function foo(){

　　　　　　myLib.doSomething();

　　　　}

　　　　return {

　　　　　　foo : foo

　　　　};

　　});

当require()函数加载上面这个模块的时候，就会先加载myLib.js文件。

## 六、加载非规范的模块

理论上，require.js加载的模块，必须是按照AMD规范、用define()函数定义的模块。但是实际上，虽然已经有一部分流行的函数库（比如jQuery）符合AMD规范，更多的库并不符合。那么，require.js是否能够加载非规范的模块呢？

回答是可以的。

这样的模块在用require()加载之前，要先用require.config()方法，定义它们的一些特征。

举例来说，underscore和backbone这两个库，都没有采用AMD规范编写。如果要加载它们的话，必须先定义它们的特征。

　　require.config({

　　　　shim: {  
  
　　　　　　'underscore':{  
　　　　　　　　exports: '\_'  
　　　　　　},

　　　　　　'backbone': {  
　　　　　　　　deps: ['underscore', 'jquery'],  
　　　　　　　　exports: 'Backbone'  
　　　　　　}

　　　　}

　　});

require.config()接受一个配置对象，这个对象除了有前面说过的paths属性之外，还有一个shim属性，专门用来配置不兼容的模块。具体来说，每个模块要定义（1）exports值（输出的变量名），表明这个模块外部调用时的名称；（2）deps数组，表明该模块的依赖性。

比如，jQuery的插件可以这样定义：

　　shim: {

　　　　'jquery.scroll': {

　　　　　　deps: ['jquery'],

　　　　　　exports: 'jQuery.fn.scroll'

　　　　}

　　}

## 七、require.js插件

require.js还提供一系列[插件](https://github.com/jrburke/requirejs/wiki/Plugins" \t "http://www.ruanyifeng.com/blog/2012/11/_blank)，实现一些特定的功能。

domready插件，可以让回调函数在页面DOM结构加载完成后再运行。

　　require(['domready!'], function (doc){

　　　　// called once the DOM is ready

　　});

text和image插件，则是允许require.js加载文本和图片文件。

　　define([

　　　　'text!review.txt',

　　　　'image!cat.jpg'

　　　　],  
  
　　　　function(review,cat){

　　　　　　console.log(review);

　　　　　　document.body.appendChild(cat);

　　　　}

　　);

类似的插件还有json和mdown，用于加载json文件和markdown文件。

# Code

Arcgis.jsp

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<%@ include file=*"/common/easyUI.jsp"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<!--<!DOCTYPE html>-->

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"*>

<meta name=*"viewport"* content=*"initial-scale=1, maximum-scale=1,user-scalable=no"*/>

<title>Simple Map</title>

<!-- <link rel="stylesheet" href="https://js.arcgis.com/3.18/esri/css/esri.css"> -->

<link rel=*"stylesheet"* type=*"text/css"* href=*"http://192.168.1.222:8888/arcgis\_js\_api\_318/library/3.18/3.18/dijit/themes/tundra/tundra.css"*/>

<link rel=*"stylesheet"* href=*"http://192.168.1.222:8888/arcgis\_js\_api\_318/library/3.18/3.18/dijit/themes/nihilo/nihilo.css"*>

<link rel=*"stylesheet"* type=*"text/css"* href=*"http://192.168.1.222:8888/arcgis\_js\_api\_318/library/3.18/3.18/esri/css/esri.css"* />

<style>

**html,** **body,** *#map* {

height: *100%*;

margin: *0*;

padding: *0*;

}

*#BasemapToggle* {

position: *absolute*;

top: *40px*;

right: *25px*;

z-index: *50*;

}

*#info* {

top: *40px*;

color: *#444*;

overflow: *hidden*;

font-family: *arial*;

right: *110px*;

/\* height: 30px; \*/

/\* width: 115px; \*/

/\* margin: 5px ; \*/

padding: *5px*;

position: *absolute*;

z-index: *40*;

/\* border: solid 2px #666;

border-radius: 4px;

background-color: #fff; \*/

}

*#pointInfo* {

bottom: *5px*;

right: *20px*;

margin: *5px*;

padding: *2px 5px*;

position: *absolute*;

z-index: *40*;

border-radius: *4px*;

background-color: *#fff*;

}

**button** {

display: *block*;

height: *36px*;

margin-left: *10px*;

float: *left*;

background: *rgb(201,233,255)*;

/\* margin: 2px 2px;

width: 26px;

position: relative; \*/

}

*#ui-esri-dijit-geocoder* {

top: *20px*;

left: *70px*;

position: *absolute*;

z-index: *3*;

}

</style>

<script type=*"text/javascript"* src=*"http://192.168.1.222:8888/arcgis\_js\_api\_318/library/3.18/3.18/init.js"*></script>

<script src=*"camerainfo.js"*></script>

<!-- 预览视频功能 -->

<script type=*"text/javascript"* src=*"preview.js"*></script>

<script type=*"text/javascript"* src=*"measure.js"*></script>

<!-- <script src="https://js.arcgis.com/3.18/"></script> -->

<script>

//声明map

**var** map,graphicLayer;

**var** geometryService;

//标记数组

**var** allMarkers = [];

require(["esri/map",

"esri/dijit/BasemapToggle",

"esri/toolbars/draw",

"esri/dijit/Geocoder",

"esri/symbols/SimpleMarkerSymbol",

"esri/symbols/PictureMarkerSymbol",

"esri/symbols/SimpleLineSymbol",

"esri/symbols/PictureFillSymbol",

"esri/symbols/CartographicLineSymbol",

"esri/InfoTemplate",

"esri/graphic",

"esri/layers/GraphicsLayer",

"esri/geometry/Point",

"esri/SpatialReference",

"esri/tasks/LengthsParameters",

"esri/tasks/GeometryService",

"esri/Color", "dojo/dom", "dojo/on",

"dojo/domReady!"],

**function**(Map,

BasemapToggle,

Draw,Geocoder,

SimpleMarkerSymbol,

PictureMarkerSymbol,

SimpleLineSymbol,

PictureFillSymbol,

CartographicLineSymbol,

InfoTemplate,

Graphic,

GraphicsLayer,

Point,

SpatialReference,

LengthsParameters,

GeometryService,

Color,

dom,

on) {

//测量Service arcgis（官网）外网 GeometryService服务地址

geometryService = **new** GeometryService("https://utility.arcgisonline.com/ArcGIS/rest/services/Geometry/GeometryServer");

//海洋渔业局-一张图-专网（内网） GeometryServer服务地址http://10.128.4.39:6080/arcgis/rest/services/Utilities/Geometry/GeometryServer

//geometryService = new GeometryService("http://10.128.4.39:6080/arcgis/rest/services/Utilities/Geometry/GeometryServer");

//new map对象

map = **new** Map("map", {

basemap: "osm",

center: [120.15, 30.26],

nav:**true**,//8个pan 箭头

slider:**false**,//左上的缩放 +/-;

logo:**false**,//右下的esri logo

showAttribution:**false**,//右下的gisNeu (logo左侧)

zoom: 10

});

//map.setExtent(map.extent.expand(0.5));

//地图缩放响应事件: zoom大于9 显示camera 否则隐藏

map.on("zoom-end", **function**(e) {

**var** zoom = map.getZoom();

//console.info(map.graphics);

**var** cameraLayer = map.getLayer("cameralayer");

console.info(cameraLayer.graphics);

**if**(zoom<9){ //zoom小于8时隐藏camare的显示

cameraLayer.setVisibility(**false**);

}**else**{

cameraLayer.setVisibility(**true**);

}

});

//dynamicMapServiceLayer 添加图层

/\* var dynamicMapServiceLayer = new ArcGISDynamicMapServiceLayer("https://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI\_Population\_World/MapServer", {

"opacity" : 1

});

map.addLayer(dynamicMapServiceLayer);

\*/

**var** geocoder = **new** Geocoder({

arcgisGeocoder: {

placeholder: "Search "

},

map: map

}, "ui-esri-dijit-geocoder");

geocoder.startup();

//鼠标样式

/\* map.setMapCursor("help"); //

map.setMapCursor("default");//

map.setMapCursor("pointer");//

map.setMapCursor("wait");//

map.setMapCursor("progress");//

map.setMapCursor("cell");// 粗十字

map.setMapCursor("crosshair");// 细十字

map.setMapCursor("text");//文本I

map.setMapCursor("vertical-text");//放倒的I \*/

//创建图层

graphicLayer = **new** GraphicsLayer({"id":"cameralayer"});

//把图层添加到地图上

map.addLayer(graphicLayer);

map.on("load", initToolbar);

**var** markerSymbol = **new** SimpleMarkerSymbol();

markerSymbol.setPath("M16,4.938c-7.732,0-14,4.701-14,10.5c0,1.981,0.741,3.833,2.016,5.414L2,25.272l5.613-1.44c2.339,1.316,5.237,2.106,8.387,2.106c7.732,0,14-4.701,14-10.5S23.732,4.938,16,4.938zM16.868,21.375h-1.969v-1.889h1.969V21.375zM16.772,18.094h-1.777l-0.176-8.083h2.113L16.772,18.094z");

markerSymbol.setColor(**new** Color("#00FFFF"));

//标注图片

**var** p\_symbol = **new** PictureMarkerSymbol({

"url":"qiangji.png",

"height":32,

"width":32

});

p\_symbol.setOffset(18, 12);

**var** p\_symbol1 = **new** PictureMarkerSymbol({

"url":"qiuji.png",

"height":32,

"width":32

});

**var** lineSymbol = **new** CartographicLineSymbol(

CartographicLineSymbol.STYLE\_SOLID,

**new** Color([0,0,0]), 2,

CartographicLineSymbol.CAP\_ROUND,

CartographicLineSymbol.JOIN\_ROUND, 5

);

**var** fillSymbol = **new** PictureFillSymbol(

"qiangji.png",

**new** SimpleLineSymbol(

SimpleLineSymbol.STYLE\_SOLID,

**new** Color('#000'),

1

), 42, 42);

**function** initToolbar() {

tb = **new** Draw(map);

tb.on("draw-end", addGraphic);

on(dom.byId("info"), "click", **function**(evt) {

$("#measure").css("display","none");//隐藏测距显示内容

**if** ( evt.target.id === "info" ) {

**return**;

}

//删除已有的图形

map.graphics.clear();

map.setMapCursor("crosshair");//鼠标形状为十字

//激活画图工具

**var** tool = evt.target.id.toLowerCase();

map.disableMapNavigation();

tb.activate(tool);

});

}

//画图

**function** addGraphic(evt) {

tb.deactivate();

map.enableMapNavigation();

**var** symbol;

**if** ( evt.geometry.type === "point" || evt.geometry.type === "multipoint") {

symbol = p\_symbol;//markerSymbol;

} **else** **if** ( evt.geometry.type === "line" || evt.geometry.type === "polyline") {

symbol = lineSymbol;//lineSymbol;

map.graphics.add(**new** Graphic(evt.geometry, symbol));

measuregeometry = evt.geometry;

MeasureGeometry(evt.geometry);//调用测量方法

map.setMapCursor("default");//

}

**else** {

symbol = lineSymbol;//fillSymbol;

map.graphics.add(**new** Graphic(evt.geometry, symbol));

showResults(evt);//播放框选点位

}

}

//播放框选出来的点位监控

**function** showResults(evt){

**var** graphics = map.getLayer("cameralayer").graphics;

**var** graphicCamera = [];

**for**(**var** i= 0, total=graphics.length;i<total;i++){

**if**(evt.geometry.contains(graphics[i].geometry)){

graphicCamera.push(graphics[i].attributes.indexCode);

}

}

**if**(graphicCamera.length > 0){

$('#videoDialog').dialog({

autoOpen : **false**,

modal : **true**,

maximizable:**true**,

width : 600,

height : 500,

draggable : **true**,

onBeforeOpen:**function**(){

**for**(**var** i = 0;i<graphicCamera.length;i++){

**if**(graphicCamera.length == 1){

ocx.setWindowsLayout(1);//设置播放插件窗口

}**else** **if**(graphicCamera.length < 5 && graphicCamera.length > 1){

ocx.setWindowsLayout(2);

}**else** **if**(graphicCamera.length < 10 && graphicCamera.length > 4){

ocx.setWindowsLayout(3);

}**else**{

ocx.setWindowsLayout(0);

}

playCameraVideo(graphicCamera[i]);

}

},

onMaximize : **function**(){

},

onBeforeClose : **function**(){

ocx.setSelectWindow(0);

stopAll();

map.graphics.clear();

map.setMapCursor("default");

}

});

}

}

**function** launchFullScreen(element) {

**if** (element.requestFullscreen) {

element.requestFullscreen();

} **else** **if** (element.mozRequestFullScreen) {

element.mozRequestFullScreen();

} **else** **if** (element.webkitRequestFullscreen) {

element.webkitRequestFullscreen();

} **else** **if** (element.msRequestFullscreen) {

element.msRequestFullscreen();

}

}

//切换地图模式

**var** toggle = **new** BasemapToggle({

map: map,

basemap: "satellite"

}, "BasemapToggle");

toggle.startup();

//添加标注

addMarker(120.15, 30.26);

// setMapCenter(120.15, 30.26,10);

//鼠标移动 显示坐标

dojo.connect(map, "onMouseMove", showCoordinates);

map.on("load", **function**() {//图形鼠标点击响应事件

//map.getLayer("cameralayer") 获取放置camera的graphicLayer

map.getLayer("cameralayer").on("dbl-click",**function**(e){

**var** cameraInfo = e.graphic.attributes;

e.stopPropagation();

openDialogVideo(cameraInfo.indexCode);

});

});

**function** showCoordinates(evt) {

//get mapPoint from event

//The map is in web mercator - modify the map point to display the results in geographic

**var** mp = esri.geometry.webMercatorToGeographic(evt.mapPoint);

//display mouse coordinates

dojo.byId("pointInfo").innerHTML = mp.x.toFixed(5) + ", " + mp.y.toFixed(5);

}

});

//打开dialog并播放视频

**function** openDialogVideo(indexCode){

**var** videoDialog = $('#videoDialog').dialog({

autoOpen : **false**,

modal : **true**,

maximizable:**true**,

width : 600,

height : 500,

draggable : **true**,

onBeforeOpen:**function**(){

ocx.setWindowsLayout(1);

playCameraVideo(indexCode);

},

onMaximize : **function**(){

},

onBeforeClose : **function**(){

stopAll();

}

});

}

//打开摄像头视频流

**function** playCameraVideo(indexcode){

//ocx.setWindowsLayout(WindowsLayout);

$.ajax({

url : "../../../video/ws/getStream",

type : "post",

data : {

indexCode : indexcode

},

dataType : "text",

success : **function**(xml) {

**var** winindex = ocx.getSelectWindow();

**if**(ocx.startPreview(winindex, xml)==0){

**if**(ocx.setSelectWindow(winindex+1)==-1){

ocx.setSelectWindow(0);

}

}

indexcodes[winindex] = indexcode;

}

});

}

//设置对中点

**function** setMapCenter(xx, yy , level) {

**var** point = **new** esri.geometry.Point(xx, yy, map.spatialReference);

map.centerAndZoom(point, level);

}

//获取点位信息,同时标注在页面中

**function** getPointInfo(){

$.post(path+'/arcgis/cameraInfo/getCamera',{"operaId":"ssjk"},**function**(json){

**var** result = eval("("+ json +")");

**var** cameraInfos = result.rows;

**for**(**var** i= 0;i<cameraInfos.length;i++){

//确定点位坐标

**var** pt = **new** esri.geometry.geographicToWebMercator(**new** esri.geometry.Point({

"x": cameraInfos[i].longitude,

"y": cameraInfos[i].latitude,

"spatialReference": { "wkid": 102113 }

}));

//确定camera类型

**var** cameratype = "qiangji.png";//枪机

**if**(cameraInfos[i].cameraType == "1"){

cameratype = "qiuji.png";//球机

}

//设置标注显示的图标

**var** symbolP = **new** esri.symbol.PictureMarkerSymbol(cameratype, 32, 32);

symbolP.setOffset(16, 16);

**var** attr = cameraInfos[i];

**var** content = "<b>名称</b>：" +cameraInfos[i].name

+ "<br><b>"+ channelNo +"</b>：" + cameraInfos[i].channelNo

+ "<br><br><a href='#' onclick='openDialogVideo(\""+ cameraInfos[i].indexCode +"\")'>播放</a>";

**var** infoTemplate = **new** esri.InfoTemplate("摄像头信息", content);

**var** graphic = **new** esri.Graphic(pt,symbolP);//,attr,infoTemplate

graphic.setAttributes(attr);

graphic.setInfoTemplate(infoTemplate);

graphicLayer.add(graphic);

}

});

}

//添加标注方法

**function** addMarker(xx, yy) {

getPointInfo();

//设置标注的经纬度

//var pt = new esri.geometry.Point(xx, yy, map.spatialReference);

//方法二

**var** pt = **new** esri.geometry.geographicToWebMercator(**new** esri.geometry.Point({

"x": xx,

"y": yy,

"spatialReference": { "wkid": 102113 }

}));

//设置标注显示的图标

**var** symbolP = **new** esri.symbol.PictureMarkerSymbol("qiangji.png", 32, 32);

symbolP.setOffset(16, 16);

**var** symbolP2 = **new** esri.symbol.PictureMarkerSymbol("qiuji.png", 32, 32);

symbolP.setOffset(0, 16);

//要在模版中显示的参数

**var** attr = {"add":"杭州市","status":"在线","attributes":{"indexCode":"001073"}};

//创建模版

**var** infoTemplate = **new** esri.InfoTemplate("标题", "地址: ${add} <br/> 状态: ${status}");

//创建图像

**var** graphic = **new** esri.Graphic(pt,symbolP,attr,infoTemplate);

//把图像添加到刚才创建的图层上

//graphicLayer.add(graphic);

/\* var graphic2 = new esri.Graphic(pt1, symbolP2, attr, infoTemplate);

graphicLayer.add(graphic2);

var graphic3 = new esri.Graphic(pt2, symbolP, attr, infoTemplate);

graphicLayer.add(graphic3);

var graphic4 = new esri.Graphic(pt3, symbolP2, attr, infoTemplate);

graphicLayer.add(graphic4); \*/

}

</script>

</head>

<body class=*"easyui-layout"*>

<div data-options=*"region:'center'"*>

<div id=*"info"* style="display: *block*;">

<button id=*"Polyline"*>测距</button>

<button id=*"Extent"*>框选</button>

<!-- <div>Select a shape then draw on map to add graphic</div>

<button id="Point">Point</button>

<button id="Multipoint">Multipoint</button>

<button id="Line">Line</button>

<button id="Polyline">Polyline</button>

<button id="FreehandPolyline">Freehand Polyline</button>

<button id="Polyline">Polyline</button>

<button id="Triangle">Triangle</button>

<button id="Extent">Rectangle</button>

<button id="Circle">Circle</button>

<button id="Ellipse">Ellipse</button>

<button id="Polygon">Polygon</button>

<button id="FreehandPolygon">Freehand Polygon</button> -->

</div>

<div id=*"measure"* style="display: *none*;">

<div id=*"result"* style="float: *left*;" ></div>

<div id=*"infoclose"* style="float: *right*;"><img alt=*"关闭"* width=*"12"* height=*"12"* src=*"close.png"*></div>

</div>

<div id=*"pointInfo"*></div>

<!-- <div id="ui-esri-dijit-geocoder"></div> -->

<div id=*"map"*>

<div id=*"BasemapToggle"*></div>

</div>

<div style="display: *none*;">

<div id=*"videoDialog"* class=*"dialog"* title= *"视频播放"* >

<div class=*"win\_container"* data-index=*"1"* id=*"preview\_1"* style="width: *100%*; height: *100%*;" name=*"ocx\_1"*>

摄像头信息

</div>

</div>

</div>

<script type=*"text/javascript"* src=*"realtimeplay.js"*></script>

</div>

</body>

</html>

Measure.js

//量算

**function** doMeasure(geometry) {

//更加类型设置显示样式

measuregeometry = geometry;

toolbar.deactivate();

**switch** (geometry.type) {

**case** "polyline":

**var** symbol = **new** esri.symbol.SimpleLineSymbol(esri.symbol.SimpleLineSymbol.STYLE\_SOLID, **new** dojo.Color([0, 0, 0]), 2);

**break**;

**case** "polygon":

**var** symbol = **new** esri.symbol.SimpleFillSymbol(esri.symbol.SimpleFillSymbol.STYLE\_NONE, **new** esri.symbol.SimpleLineSymbol(esri.symbol.SimpleLineSymbol.STYLE\_DASHDOT, **new** dojo.Color([255, 0, 0]), 2), **new** dojo.Color([255, 255, 0, 0.25]));

**break**;

}

//设置样式

**var** graphic = **new** esri.Graphic(geometry,symbol);

//清除上一次的画图内容

map.graphics.clear();

map.graphics.add(graphic);

//进行投影转换，完成后调用projectComplete

MeasureGeometry(geometry);

}

//投影转换完成后调用方法

**function** MeasureGeometry(geometry) {

//如果为线类型就进行lengths距离测算

**if** (geometry.type == "polyline") {

**var** lengthParams = **new** esri.tasks.LengthsParameters();

lengthParams.polylines = [geometry];

lengthParams.lengthUnit = esri.tasks.GeometryService.UNIT\_METER;

lengthParams.geodesic = **true**;

lengthParams.polylines[0].spatialReference = **new** esri.SpatialReference(102113);

geometryService.lengths(lengthParams);

dojo.connect(geometryService, "onLengthsComplete", outputDistance);

}

//如果为面类型需要先进行simplify操作在进行面积测算

**else** **if** (geometry.type == "polygon") {

**var** areasAndLengthParams = **new** esri.tasks.AreasAndLengthsParameters();

areasAndLengthParams.lengthUnit = esri.tasks.GeometryService.UNIT\_METER;

areasAndLengthParams.areaUnit = esri.tasks.GeometryService.UNIT\_SQUARE\_METERS;

**this**.outSR = **new** esri.SpatialReference({ wkid: 102113 });

geometryService.project([geometry], **this**.outSR, **function** (geometry) {

geometryService.simplify(geometry, **function** (simplifiedGeometries) {

areasAndLengthParams.polygons = simplifiedGeometries;

areasAndLengthParams.polygons[0].spatialReference = **new** esri.SpatialReference(102113);

geometryService.areasAndLengths(areasAndLengthParams);

});

});

dojo.connect(geometryService, "onAreasAndLengthsComplete", outputAreaAndLength);

}

}

//显示测量距离

**function** outputDistance(result) {

**var** CurX = measuregeometry.paths[0][measuregeometry.paths[0].length - 1][0];

**var** CurY = measuregeometry.paths[0][measuregeometry.paths[0].length - 1][1];

**var** CurPos = **new** esri.geometry.Point(CurX, CurY, map.spatialReference);

measureInfo(CurPos,result.lengths[0].toFixed(3), "米");

/\*map.infoWindow.setTitle("距离测量");

map.infoWindow.setContent(" 测 量 长 度 ： <strong>" + parseInt(String(result.lengths[0])) + "米</strong>");

map.infoWindow.show(CurPos);\*/

}

//显示测量面积

**function** outputAreaAndLength(result) {

**var** CurX = (measuregeometry.\_extent.xmax + measuregeometry.\_extent.xmin) / 2;

**var** CurY = (measuregeometry.\_extent.ymax + measuregeometry.\_extent.ymin) / 2

**var** CurPos = **new** esri.geometry.Point(CurX, CurY, map.spatialReference);

map.infoWindow.setTitle("面积测量");

map.infoWindow.setContent(" 面积 ： <strong>" + parseInt(String(result.areas[0])) + "平方米</strong> 周长：" + parseInt(String(result.lengths[0])) + "米");

map.infoWindow.show(CurPos);

//alert("面积：" + dojo.number.format(result.areas[0]) + "平方米" + " 长度：" + dojo.number.format(result.lengths[0]) + "米");

}

/\*\*

\* 显示测量结果

\* **@param** showPnt

\* **@param** data

\* **@param** unit

\*/

**function** measureInfo(showPnt,data,unit){

**var** measureDiv=$("#measure");

**var** infoCloseDiv = $("#infoclose");

**var** isShow = **false**;

**var** screenPnt=map.toScreen(showPnt);

measureDiv.css("left",screenPnt.x+"px");

measureDiv.css("top",screenPnt.y+"px");

measureDiv.css("position","absolute");

measureDiv.css("height","26px");

measureDiv.css("line-height","26px");

measureDiv.css("padding-left","5px");

measureDiv.css("background-color","#FFF");

measureDiv.css("border-radius","4px");

measureDiv.css("border","1px solid red");

measureDiv.css("display","block");

infoCloseDiv.css("margin","1px 1px");

infoCloseDiv.css("margin-left","5px");

infoCloseDiv.css("cursor","pointer");

isShow = **true**;

measureDiv.css("z-index","999");

**if**(unit==="米"){

//measureDiv.css("width","120px");

}

**else**{

//measureDiv.css("width","130px");

}

$("#result").html(data+" "+unit);

$("#infoclose").click(**function**(){

map.graphics.clear();

map.setMapCursor("default");

measureDiv.css("display","none");

isShow = **false**;

});

map.on("pan-start", **function**(){

measureDiv.css("display","none");

});

map.on("pan-end", **function**(panend){

**if**(isShow == **true**){

screenPnt=map.toScreen(showPnt);

measureDiv.css("left",screenPnt.x+"px");

measureDiv.css("top",screenPnt.y+"px");

measureDiv.css("position","absolute");

measureDiv.css("height","26px");

measureDiv.css("line-height","26px");

measureDiv.css("padding-left","5px");

measureDiv.css("background-color","#FFF");

measureDiv.css("border-radius","4px");

measureDiv.css("border","1px solid red");

measureDiv.css("display","block");

infoCloseDiv.css("margin","1px 1px");

infoCloseDiv.css("margin-left","5px");

infoCloseDiv.css("cursor","pointer");

}

});

map.on("zoom-start", **function**(){

measureDiv.css("display","none");

});

map.on("zoom-end", **function**(){

**if**(isShow == **true**){

screenPnt=map.toScreen(showPnt);

measureDiv.css("left",screenPnt.x+"px");

measureDiv.css("top",screenPnt.y+"px");

measureDiv.css("position","absolute");

measureDiv.css("height","26px");

measureDiv.css("line-height","26px");

measureDiv.css("padding-left","5px");

measureDiv.css("background-color","#FFF");

measureDiv.css("border-radius","4px");

measureDiv.css("border","1px solid red");

measureDiv.css("display","block");

infoCloseDiv.css("margin","1px 1px");

infoCloseDiv.css("margin-left","5px");

infoCloseDiv.css("cursor","pointer");

}

});

};