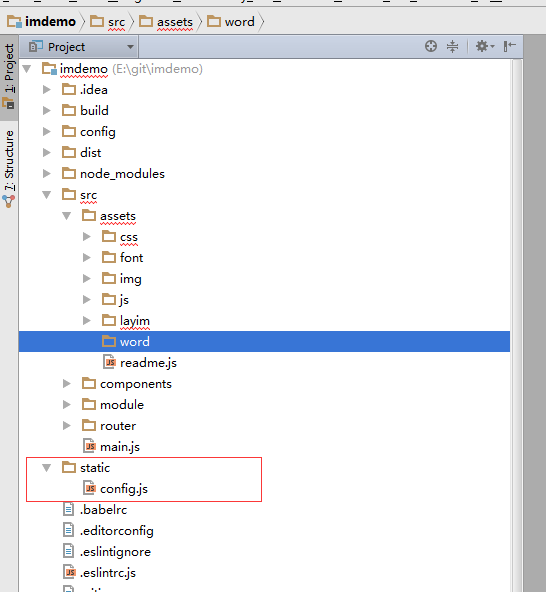
Webpack部署在二级或者多级目录下的操作

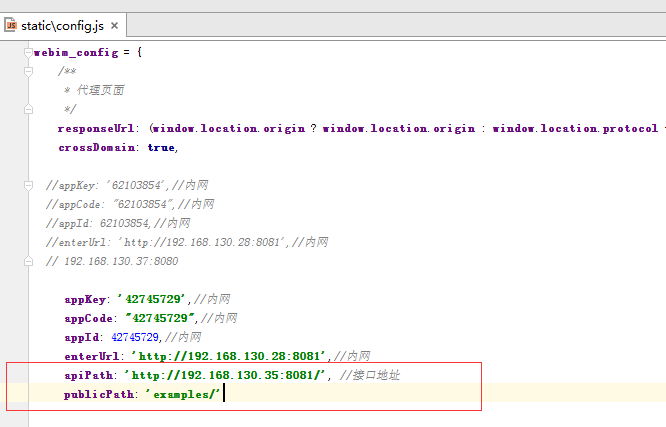
## 预留出配置文件config.js

这个是二级部署的核心之一。

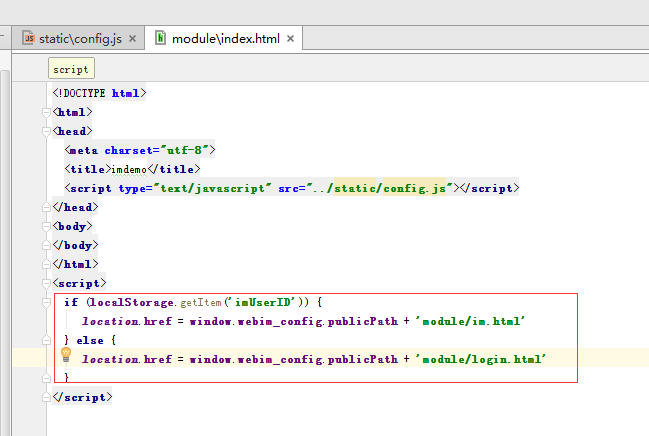
* 把Config.js放置在static目录下面



* 在Config.js中标记二级目录的路径

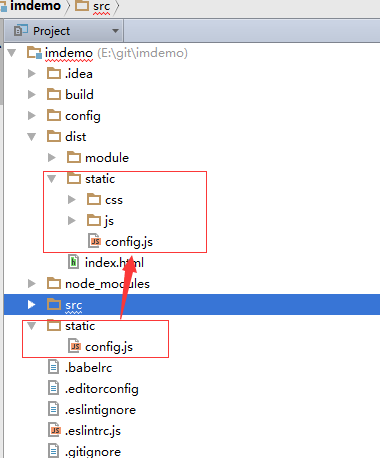


* 所有跳转的地方都注明这个前缀，此处仅列出一个示例



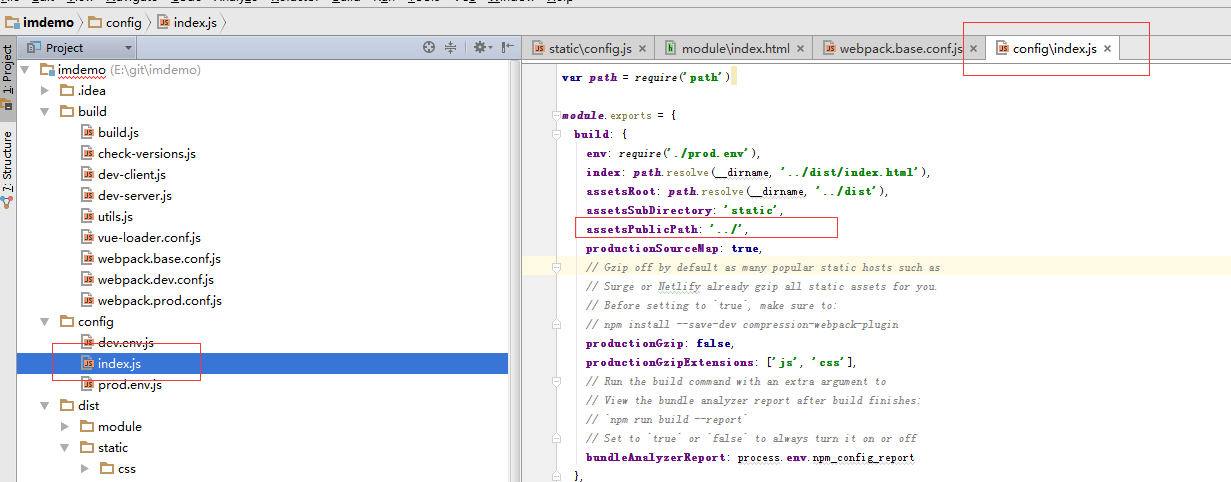
* 所有的html页面都需要单独引入config.js文件，而且这个引入必须放在其他script引用的前面，建议放在head中，因为业务需要，其他的js代码可能需要config.js中的内容，此时config.js的路径由在static中放置的路径决定，具体查看在npm run build之后的dist文件夹下





## 如何设置项目放置在二级目录下

如果是根目录，config/index.js内的assetsPublicPath：‘/’，如果是二级，则为’../’



## 如何处理项目在二级目录下，引入的css文件中如果嵌套了图片，字体等文件的引入，路径会出错

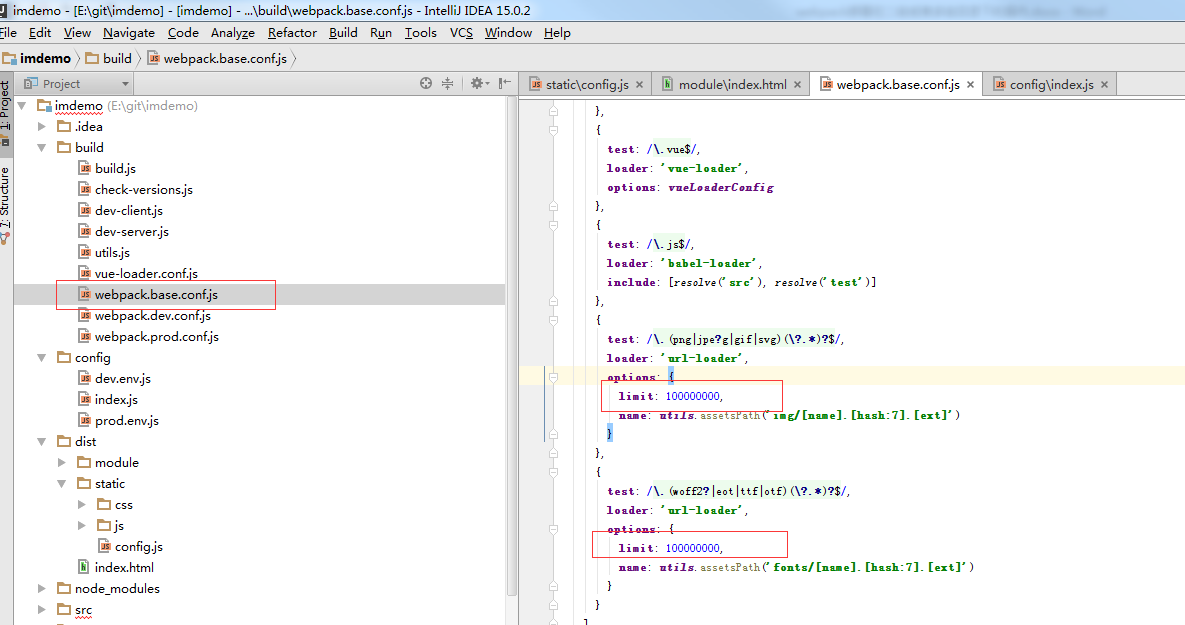


有三种做法：

### 第一种

改变打包的时候图片和字体的limit值，如果limit为默认的10000，则表示图片和字体小于10kb的转换成base64编码，否则转为文件，因为在加载页面的时候，多图片浏览器会开启多个下载线程，如果图片很小，很影响性能，故而转为base64，单一线程下载，但是如果是大图片，则应该是多开线程，而不是转为base64，因为这样单一文件过大，也造成阻塞，影响性能。10kb是最好的分割线。

这里调大limit，就可以在加载对应的后缀文件的时候，转为base64，这样就解决路径问题了。



### 第二种

Cdn处理，就是类似图片，字体这样的静态文件，统一放置在cdn中，css中引用的都是cdn的地址，这样其实是最优的选择，但是弊端是要通篇更改css中，甚至是代码中的文件引用路径为cdn路径，有工作量

### 第三种

略