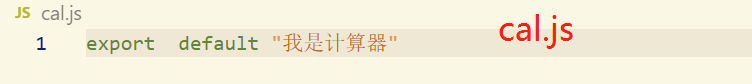
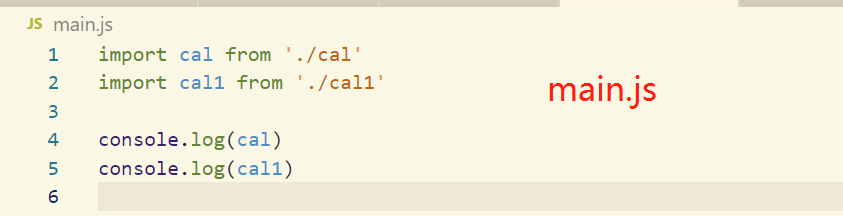
编写最简单的模块化代码

1



2.



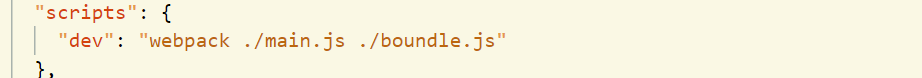
3.



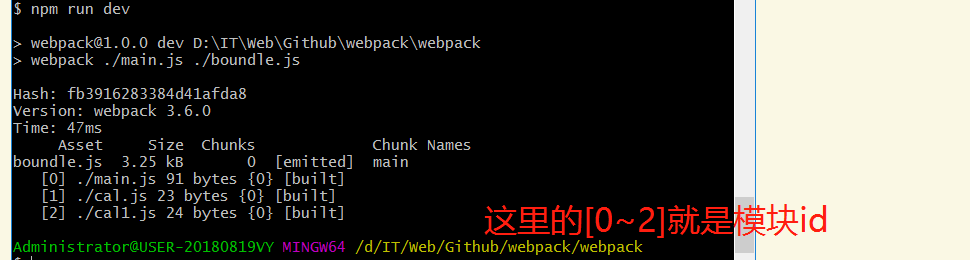
4.



5.执行npm run dev 就是在执行



执行结果如下所示



这里的[0] 表示main.js 的模块id就是0

看看打包后的boundle.js文件

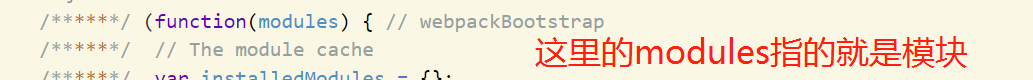
(function(moudle){

…..一堆代码

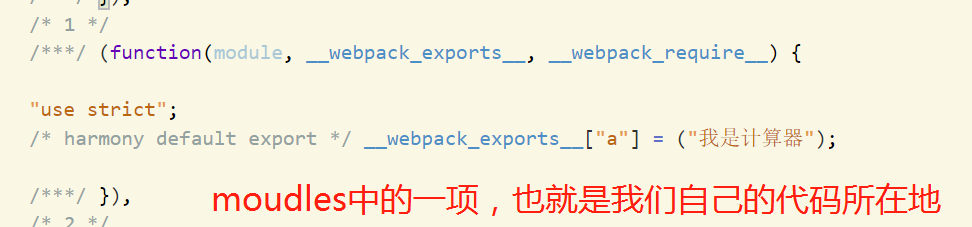
}) ([fn1,fn2,fh3])

整个boundle.js其实就是个一个立即执行的函数，加载完就执行完了

其中



这里的modules其实是个数组[fn1,fn2,fn3],数组中的每一项都是一个函数

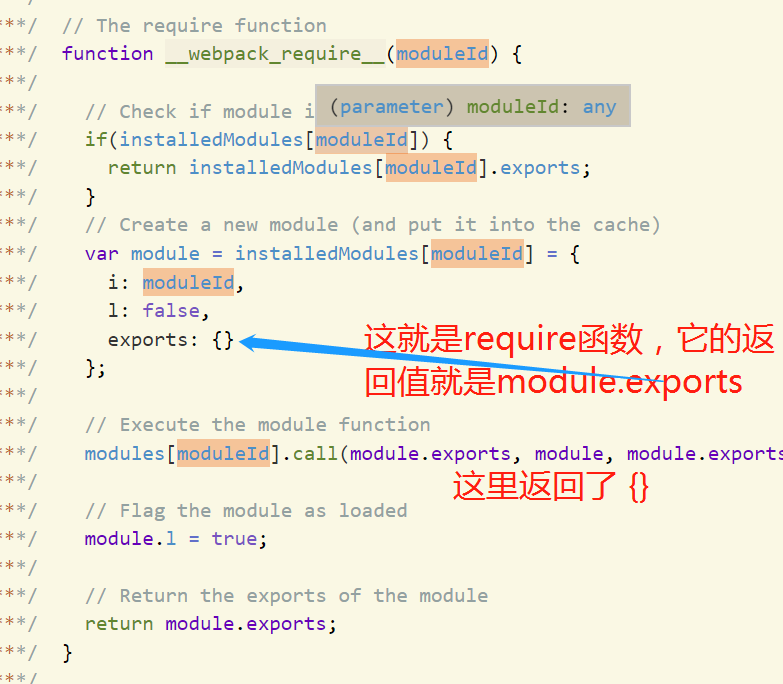


数组中每一项对应的函数的参数





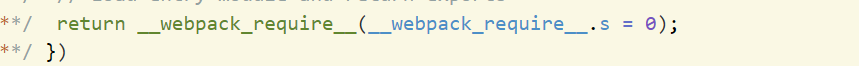
这里的moduleId 就是指模块的id



返回的是个空对象



最终我们看到主函数，立即执行函数最终的返回值是

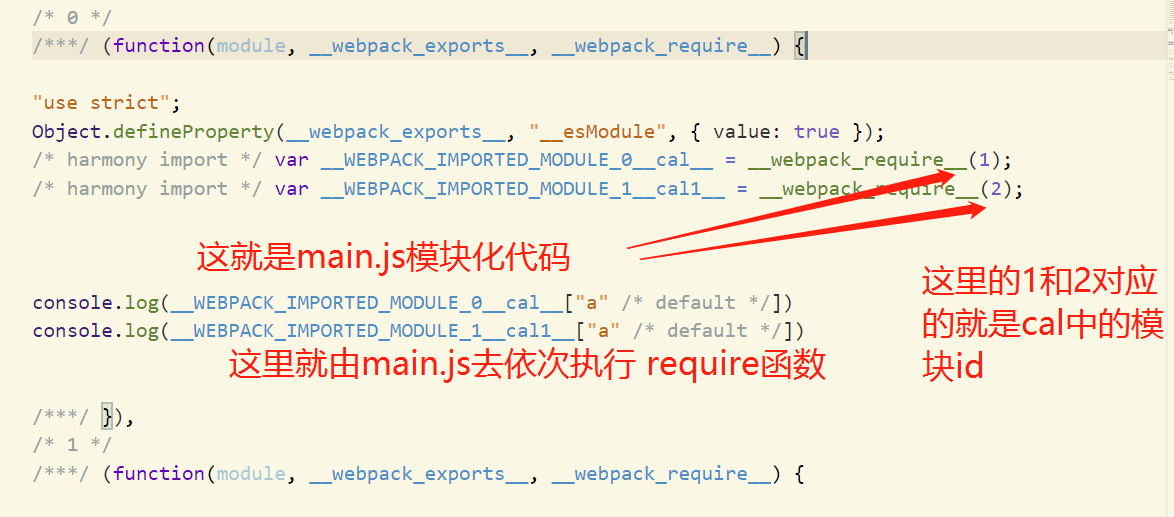


这个等价于

return \_\_webpack\_require\_\_(0)

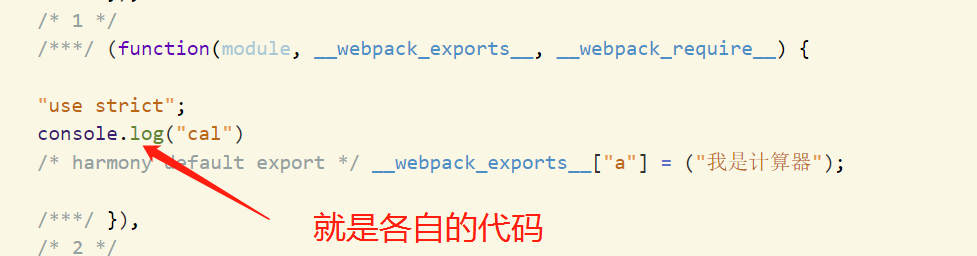
意思是执行 执行模块id为0的modules中的函数，其实就是第一项中的函数

也就是main.js的模块



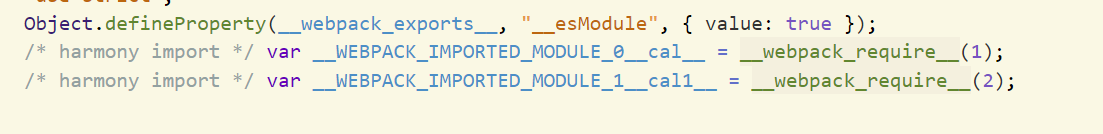
意思是执行了main.js 这个入口文件的代码时，已经调用的其所依赖的模块代码

立即执行函数-----模块函数代码参数-----执行入口文件模块代买----执行其依赖模块代码----执行各自代码



这里webpack其实就是将各个模块的代码先进行一个组合----modules的参数，

将这些模块化的代码变成一个函数，通过函数调函数的方式去执行每个模块中相应的代码，在这里当\_\_webpack\_require\_\_(1)





执行此函数，就执行了cal模块当中的代码，

通俗易懂的说法就是：

Webpack将你模块的代码全部转变成一个个可执行的函数，在入口文件执行时，依次调用所依赖的所有模块的函数，进行全部调用，